(2)

Final report on research done under the project:

Database Explorer: Large scale database application

Supported by:

Manpower, Personnel, and Training R&D Program
Office of Naval Research

Principal Investigator: Jan M. Żytkow
phone: (316) 689-3925
e-mail: zytkow@wise.cs.twsu.edu

ONR Grant No: #N00014-91-J-1362

Research Conducted at: *Wichita State University*
Department of Computer Science
Wichita, KS 67208

Project Period: Jan.25, 1991 – Jan.24, 1992

Executive Summary

Large databases maintained by the Navy are a potential source of useful knowledge. Yet this knowledge is only implicit in the data. It must be mined and expressed in a concise, useful form of statistical patterns, equations, rules, concepts, and the like. Automation of knowledge mining is important because databases are very large, numerous, and rapidly growing.

In the previous project (ONR N00014-90-J-1603; March 1, 1990 – March 31, 1991) we have developed Forty-Niner (49er), a general-purpose database mining system, which conducts large-scale search for useful knowledge. In the current project (Jan.25, 1991 – Jan.24, 1992) we (1) expanded 49er into a menu-driven tool which can be used by a non-programmer, (2) tested 49er on large scale, including search in two NPRDC databases, and (3) augmented 49er's search for knowledge in several ways.

The main results of 49er's search are regularities. All have a simple general form:

in a (sub)range S of data a pattern P holds,

for instance "for all data, salary of each person is a logarithm of the length of service (within a specified error)", or "for submarine recruiters who work in metropolitan areas, their time of stay at the present station correlates negatively with their success rate".

The ranges are described by conjunctions of simple conditions, for instance "submarine recruiters", or "recruiters who work in metropolitan areas". The patterns belong to two types: (1) equations, such as linear or logarithmic dependency; (2) contingency tables. Both equations and contingency tables are very useful and widely used tools in data analysis. For instance, *Statistical Concepts and Methods* by Bhattacharyya and Johnson (1986), has a good introduction to both equations and contingency tables. Gokhale and Kullback discuss contingency tables extensively in their monograph (1978). Both equations and contingency tables are concise data summaries. Both can be used for making predictions and explanations. For more details see section 2, where we define regularities, and section 5.

All statistical tools used by 49er are elementary. This is done on purpose, so that it should be easy to interpret each result. 49er's strength lies not in single statistical tools, but in using them in a large scale automated search, which can examine many thousands of hypotheses. Scope of search is very large even for small databases. Consider a database of just 10 attributes. Consider subsets of data which can be described by conjunctions of up to three conditions. For instance, we may be interested in patterns which hold for people who live in big cities, who are older than 20, and who smoke. If all conditions are binary, and if we consider only one type of 2-dimensional pattern, there is some 25,000 hypotheses possible. A typical database uses many tens of attributes, and we may be interested in many types of patterns, so the number of reasonable hypotheses can easily reach million - billions.

A well-organized, large scale search, guided by user goals, knowl[illegible]he attributes, and partial results of data analysis, is necessary to make regularity [illegible]ective. Let us contrast 49er with statistical packages, such as SPSS (SPSS Referen[illegible]0) and Lisp-Stat (Tierney, 1990; Lisp-Stat: Book Reviews, 1991), widely used in [illegible]s. The most important advantage of 49er over any statistical package is that a packa[illegible]narily a tool box, whereas 49er is primarily an integrated production line; 49er auto[illegible]ly generates

and examines large numbers of hypotheses, whereas tools available in a statistical package verify one user-generated hypothesis at a time. It is up to the package user to decide each time which attributes to try, which hypothesis, which method, and what to do next after each result is obtained.

49er could be also viewed as a toolbox since all lower-level routines for analysis, evaluation, and graphical presentation of hypotheses can be directly called by the user. But 49er can apply each tool thousands times in running one discovery search. It is true that all what 49er does *can be eventually available* in a package such as Lisp-Stat. Indeed, in addition to the box of statistical tools, Lisp-Stat is a programming environment based on lisp. So 49er, which is programmed in lisp, could be developed in Lisp-Stat. However, the implementation would take years for somebody inexperienced with search methods applied in discovery systems. Discovery search is difficult to program because it examines large hypothesis spaces, and must do it without repetitions, without skipping important hypotheses, must examine more general hypotheses before it examines their particular cases, should not try costly patterns when a simple test can tell that they do not hold, and so forth. When these requirements are satisfied the search becomes efficient, without spending weeks of cpu time on tasks which can be done in hours. Also, a well designed discovery system allows the user to spend minutes or hours on setting the system switches rather than reprogramming, which would take weeks, or even months. 49er has been developed to satisfy these requirements.

49er searches for significant statistical patterns and equations using knowledge about database and user goals (details in section 6), conducting a more costly search for equations only when data indicate a functional relationship between variables (details in section 5). 49er combines several searches, each contributing to different aspect of a regularity (details in section 4). Correspondence between the search components and the components of knowledge makes the system easy to understand, use, and expand. Regularities can be further refined by 49er, typically yielding much stronger regularities and useful concepts (details in section 7).

We discuss 49er's performance in four categories of tests. First, we applied 49er to large databases. Search in a megabyte-size database typically yields many regularities, but it would take human experts weeks to verify the optimality of 49er's results. The second type of testing aims at reproduction of human findings. Databases which have been extensively studied are rare, however. To evaluate 49er on large scale against known results, we use hide-and-seek testing on artificially created data. In the fourth approach we analyze discovery of regularities in randomly generated databases. The results are summarized in section 8.

We have applied 49er systematically on two NPRDC databases. In sections 9 and 10 we discuss a variety of results obtained from those databases.

Contents

1 Introduction

We describe Forty-Niner (49er), a computer system for automated mining useful knowledge in relational databases, we evaluate 49er on a number of tests, and we describe results of 49er's applications on two NPRDC databases. The basic form of knowledge discovered by 49er is regularities, that is patterns common in sets of data, analogous to scientific laws. In addition, 49er introduces simple concepts. We describe the general form of regularities and we analyze a variety of their types. Regularities discovered in databases can play a role analogous to scientific laws, allowing the users to make predictions, explanations, and justified decisions.

There has been considerable interest in recent years in automated methods of mining databases for useful knowledge. Two collections of papers (Piatetsky-Shapiro & Frawley 1991; Piatetsky-Shapiro 1991) provide the overview of the state of art in Knowledge Discovery in Databases (KDD). Database mining is attractive for many reasons. There are many available databases, they are simple and uniform in structure, and a considerable amount of effort has been spent in designing them and collecting useful data. Many historical records are available in the form of databases, such as stock market prices and indexes, corporate records, census data, and weather records. The automated search for regularities is particularly attractive and useful for large databases, and the same algorithms apply to all relational databases, because of their similarity in structure. Because many databases are very large, and the forthcoming databases will encompass gigabytes or even terabytes of information, knowledge extraction on that scale must be automated.

Database management systems help to efficiently retrieve facts in response to specific queries. However, it is difficult to use the traditional database management tools on discovery tasks, because discovery requires open exploration of a large-scale hypothesis space, instead of data retrieval based on a single, user-defined pattern. Even the best statistical packages (Tierney 1990) require a human to prepare a data set and to generate hypotheses, and take one hypothesis at a time. Large databases make both approaches prohibitively slow.

Typical databases vary from thousands to many millions of records, from a few to a few hundred attributes, and from two to huge numbers of attribute values. The attribute values can be boolean, symbols or numbers. To accommodate different types of data the hypothesis space must be very large and complex. Different domain knowledge and different accuracy requirements add to the complexity of the task.

A database miner must be open to many types of regularities. It must consider a huge number of data subsets. It must be systematic, yet not repetitive. It must use knowledge, be responsive to user goals, and be able to bootstrap on the earlier results. It must be able to present the results to humans in a simple convincing way. 49er has been constructed with all these tasks in mind.

Rather than building separate discovery systems to deal with the variety of databases and various exploration goals, we developed a general-purpose database mining system, a uniform representation of domain knowledge, and search control which is relatively easy to understand and manipulate. We introduced a theoretical framework which can be used to describe and to analyze the search for regularities in relational databases.

Our knowledge discovery paradigm can be summarized in a few statements:

1. Decompose discovery into several search tasks, each search governed by its own operators, heuristics, and evaluation criteria. Each search must be related to a well-defined aspect of a regularity.

2. For each search, develop an automated applicability test.

3. Distinguish between discovery of preliminary regularities, and the regularity refinement process.

4. Develop a large scale search that captures regularities in a preliminary, simple form, at a predefined level of strength. Such regularities can be discovered fast, so that a large scale search is possible.

5. Develop various mechanisms for regularity refinement. Preliminary regularities are typically manifestations of stronger regularities, which may be obtained through gradual refinements.

6. Apply regularity refinement techniques selectively because they are computationally expensive.

7. Link all searches into a multisearch system, which can examine large hypothesis spaces and can be run either automatically, or can be interrupted to permit human inspection and guidance.

8. Develop an efficient inspection mechanism, which allows the user to review the results and decide on the next step.

9. Provide the user with access to search parameters that can be adjusted to the available resources, such as the duration of search or available memory.

10. Provide the user with tools to represent simple domain knowledge and discovery goals.

11. Link domain knowledge and discovery goals to the search control.

A database search for regularities has been automated to various degree. Only a few database exploration systems, including 49er, TETRAD (Glymour et al. 1987, 1991), and Knowledge Discovery Workbench (KDW Piatetsky-Shapiro & Matheus 1991), conduct a systematic, large scale search.

KDW (Piatetsky-Shapiro & Matheus 1991) conducts a large scale exploration, using several tools: it checks functional dependence of two variables, clusters two dimensional data into a number of linear dependencies, finds application conditions for each cluster, links these conditions into a decision tree, discovers anomalous records, which do not fit the known regularities, and provides the user with a number of data and regularity visualization methods. In comparison to KDW, the scope of 49er's search is larger, both in terms of the subsets of data explored and patterns considered.

TETRAD (Glymour et al. 1987, 1991) searches for a network of causal relations between attributes. In distinction, in 49er we assume that database users know which attributes they can control, and they know their goal attributes, which they would like to influence by

manipulations on controls. The combinations of goal and control attributes determine the scope of search for useful regularities.

In a limited form of automation, the user is charged with hypotheses generation, while verification is conducted automatically (Naqvi and Tsur 1989). This approach, called *data dredging* (Chimenti et al. 1990) does not go beyond the traditional database management mode of answering user-generated queries. In contrast, Cai, Cercone and Han (1989) propose an inductive, partially automated process of generation and generalization of hypotheses. Their search is guided by concept hierarchies, one for each database attribute, and the user-specified relevance relation. The progress is evaluated based on the user-specified thresholds.

Scientific Discovery (Shrager & Langley 1990) is another main direction in Machine Discovery. The systems such as BACON (Langley et al. 1987) and FAHRENHEIT (Żytkow, 1987), use control over experiments and focus on laws derived from fine scientific data. Scientific Discovery is a useful source of solutions for KDD, but most methods must be adapted to KDD. This is because large errors, sparse data, and records with missing values are common in databases, and new data are not as readily available as in experimental sciences. Many databases describe social phenomena, not conducive to the representations useful in natural sciences. Forty-Niner applies a number of solutions from BACON and FAHRENHEIT (Żytkow & Baker 1991), but does not use the experimental approach and puts more emphasis on statistical patterns. In accord with scientific approach to data, we neglect missing values. What if frequently called "inconsistent values", in our approach is included in statistical patterns or becomes error associated with equations.

2 Overview of 49er.b

Forty-Niner.b (49er.b), described in this paper and developed in the reported period, enhances Forty-Niner (Żytkow & Baker 1991) in several ways. It can use domain knowledge to guide the search for regularities. It incorporates Equation Finder (Zembowicz & Żytkow 1991), so it is able to discover a broad range of equations. A functionality test allows 49er.b a selective use of Equation Finder on data which permits functional description. Other additions include the systematic use of chi-square test to evaluate and to rank regularities, and new mechanisms for regularity refinement.

The architecture of 49er.b is summarized in Figure 1. The system consists of two search modules. The first performs a search for two-dimensional regularities in a large hypothesis space determined by subsets of data and useful combinations of attributes. In this phase Forty-Niner covers as much data as possible with regularities. The second module refines regularities, strengthening them and/or generalizing them from two to more dimensions. Either module can be used many times, as indicated by thick arrows in Figure 1. At each cycle, the user can change search parameters and the scope of search, and continue the exploration. This architecture combines the advantages of large-scale automation and knowledge brought by a human operator. Effective visualization enables the human operator to quickly capture the specificity, promise, and importance of a given statistical pattern, and guide or supervise a more costly search for its refinement and generalization. Initially, Forty-Niner looks for simple statistical regularities, but realizing that the data follow specific patterns, the user might apply more subtle (but more costly) mechanisms, focused on the particularly

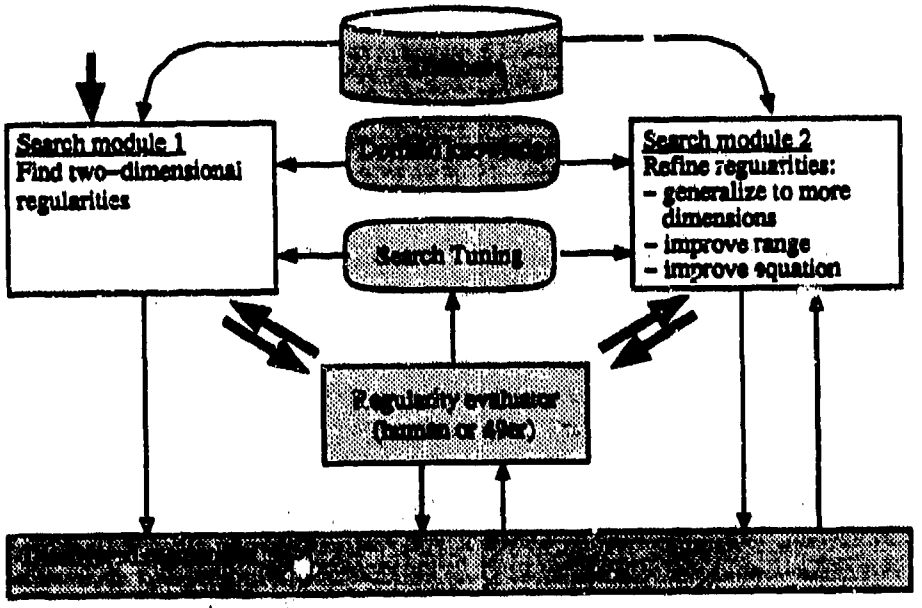

Figure 1. Overview of Forty-Niner. Thick arrows (⟹) indicate flow of control. Thin arrows (⟶) indicate flow of data.

interesting regularities.

The evaluation of regularities requires statistical tests and the selection of a threshold of acceptance for each test. Threshold selection can be either an arbitrary choice or a decision motivated by the domain knowledge and user goals. Depending on the discovered regularities, the thresholds can be adjusted to increase or decrease the number of new discoveries.

3 What Is a Regularity?

We concentrate on knowledge in the form of (1) regularities and (2) concepts in which those regularities are expressed. A *regularity* holds in a domain D if some events or combinations of events which are *a priori* possible, never occur in D, while some others can occur repeatedly. A regularity occurs when there is a pattern in states, events, processes, or the like, and not each possible event belongs to that pattern, or at least not everything is equally possible. This is probably the most general characteristics of a regularity.

The most useful special cases of regularities take on the form of equations, first order logic statements, and contingency tables. In some domains no event is excluded but some can be highly improbable. This can be expressed by statistical regularities in the form of contingency tables. In other domains the regularities are very strong and most of the events are excluded. These regularities can be expressed in the form of logic statements or mathematical equations. Equations are suitable for highly repeatable, scientific facts about numerical attributes. Equations are also useful in expressing quasi-functional relationships

when the actual values are spread within a limited variance; for instance the dependence between human weight and height, or the cost of a house on its size.

In database applications, the space of all possible events can be defined by the attributes that form a given database and their values. Given a relational database *DB* with the attributes A_1, A_2, ..., A_n, let V_1, V_2, ..., V_n be the corresponding sets of values for each attribute. The space W of all possible events is the Cartesian product $W = V_1 \times V_2 \times \ldots \times V_n$, which includes all possible combinations of values of the attributes, that is all possible events. Let $\mathcal{L}$ be the language based on these attributes and their values.

We say that a regularity R in a domain (a population) represented by a database *DB* holds in *DB* if some logically possible events never actually occur in the domain or at least not all events are equally probable. In other words, R describes a limitation in a space W of all situations (events) *a priori* possible in *DB*.

The number of possible events is usually significantly larger than the actual number of records in a typical database: for $N = 20$ attributes, each having $K = 10$ values, the number of possible events is $K^N = 10^{20}$, while the size of a typical large database is about $M = 10^6$ records. Because the actual records are so sparse, it is impossible to distinguish between an event which does not occur in *DB* because it is not possible, and event which does not occur in *DB* but is possible.

For each database *DB*, the Cartesian product W can be used to build a frequency table that shows the distribution of events (records) that actually occur in *DB*. A frequency table is a mapping from W to the set of natural numbers N, which associates each possible event $(v_1, \ldots, v_n)$ with the number of occurrences of the corresponding record in the database. Frequency tables based on W are potentially very important, because they could be viewed as a regularities. However, as we have just shown, for typical databases they are not practical, because they are very large and very sparse. For most of possible events in the Cartesian product W there are no corresponding records in the database. The sparseness of record distribution in W can be reduced by attribute projection (which reduces N) and by aggregation of attribute values (which reduces K). When projections and/or aggregations reduce the size of W to less than the number of records M, the data in the diagram become dense enough so that empty cells in the frequency table can be interpreted as impossible events. 49er applies both techniques in order to deal with statistically significant samples.

Two or more dimensional frequency tables are viewed as statistical regularities, so called contingency tables (Bhattacharyya and Johnson 1986; Gokhale and Kullback 1978). A diagram of actual frequencies can be used to reason about the domain, for instance in making predictions. Consider a two-dimensional diagram in Table 1. Suppose that we plan to select a person with the value 3 of the attribute LAS and we want to predict the values of SUCCESS. Corresponding probabilities can be computed for different combinations of values. From the diagram we can infer that value 1 of SUCCESS will occur with probability 0.28 (= 144/511), value 0.67 with probability 0.34, value 0.33 with probability 0.3, while value 0 with probability 0.08. Similar predictions can be made for any other value of LAS or for any weighted combination of values.

Predictions become unique when the diagram is a function from A_1 to A_2. For each value of A_1 they include one value of A_2 and exclude all other values. Many functions can be represented by equations, which are even more concise, easy to manipulate, and useful for making predictions and explanations. Such are the advantages of equations that even

Table 1: An example of a statistical regularity: Actual frequency grid in NPRDC's "recout.dat" database for a projection of all data to a subspace of two attributes: SUCCESS and LAS (how long at a recruiting station)

Attribute: SUCCESS

100%	20	62	144	207	162	296	378	124
67%	25	156	175	165	105	175	210	48
33%	11	35	152	62	32	53	48	9
0%	47	93	40	16	2	7	9	3
	1	2	3	4	5	6	7	8

Attribute: LAS

the more fuzzy diagrams can be represented by equations, accompanied by the values of deviation. Linear correlations capture a simple case of this idea. Although any diagram can be represented by a statement in the language $\mathcal{L}$, typically such statements would be very long, enumerating each field in the diagram. Equations, such as $A_2 = -(A_1 - a)^2 + b$ are far more concise.

Multivalued relation also exclude some events, but do not allow us to make unique predictions. Many of them can be represented in a concise first order logic form. For example, the regularity in Table 2 can be represented by the statement: *all ravens are black.*

Table 2: Black and white ravens

black	207	935
non-black	0	658
	raven	non-raven

Both functions and relationships can be called black-and-white regularities. According to them, some events can never occur in W. If all events can occur, then we have a statistical gray-level regularity, a contingency table. The predictions are the weakest when the probabilities in a given contingency table are equal. A random distribution does not allow to narrow down the scope of predicted values; each is equally probable. We may hesitate to call a random distribution a regularity, but a random world (a combination of independent variables) holds a distinct, empirically verifiable pattern. Various distributions of events in a random world are statistically improbable. Because the usefulness of random distributions is negligible, we will narrow down the notion of regularity to include only diagrams in which events occur with actual frequencies significantly different from those expected apriori.

Databases can hardly offer the quality of data available to an experimental scientist. Scientific facts are usually highly repeatable, leading to sharp, deterministic, function-type regularities described by mathematical equations. Since we cannot expect the same quality of regularities in databases, 49er searches primarily for weaker, statistical regularities. Equations are most useful for scientific databases and to express definitional relationships, which can be used as integrity constraints, but occasionally they can be used in other situations.

Regularities can be compared by their statistical strength based on various criteria. 49er uses the chi-square test and the corresponding probability that a given sample is a statistical fluctuation of random distribution. Actual probability distribution must be compared

with the expected distribution based on the null hypothesis of variables independence. The regularity can take on the form of a relative difference table (sometime they are also called contingency tables), defined by $(A_i - E_i)/E_i$, where A_i is the actual number of records in cell i, while E_i is the value based on the null hypothesis of independence applied to the histograms of the variables used in the table.

Another important characteristic of a regularity is its range: a regularity may occur in all data or in a subset of *DB*. A regularity between A_1 and A_2 may be very weak in the projection that includes all data, but in the subset of data defined by a subset of values of A_3 the regularity can be much stronger. Similarly to rules, we may expect regularities, whose range is defined by a conjunction of conditions on values of other attributes.

A regularity in a database *DB* can be expressed in the following form:

$$\text{PATTERN in RANGE,} \tag{1}$$

where RANGE defines a subset in W (whole W as a special case), while PATTERN describes the regularity that occurs for the data available in the RANGE. 49er.b finds regularities in the form:

$$\text{REG}(G, C_1, \ldots, C_l) \text{ in } P_1(S_1) \& \ldots \& P_n(S_k), \tag{2}$$

where $\{S_1, \ldots, S_k, G, C_1, \ldots, C_l\} \subseteq \{A_1, \ldots, A_N\}$; $P_i(A_i)$ is either $A_i > a_i$, $A_i \leq a_i$, or (for nominal attributes) $A_i = a_i$, a_i is a value of A_i; $\text{REG}(G, C_1, \ldots, C_k)$ is either a function $G = f(C_1, \ldots, C_k)$ or is a $k+1$ dimensional contingency table for all values of $C_1, \ldots, C_k$ and G (which we call CONTINGENCY-ALL), or is a contingency table for upper and lower values of each attribute (2^{k+1} cells, called CONTINGENCY-2). The meaning of C_i's, S_i's and G's will be explained in the next section.

Similarly to statistical literature, we use the term *contingency tables* to indicate two types ot tables: (1) frequency tables, which are similar to scatter plots. They allows to make predictions of values of one attribute based on values of other attributes. (2) difference tables, which show comparison between actual and expected frequencies of data distributions. They show unexpected events (records) in data, and details of dependency between attributes. Our CONTINGENCY-2 produces a simple and short summary, very close to the correlation coefficient, while CONTINGENCY-all shows the dependency in far greater details.

Regularities can be compared by their generality, determined by their RANGE, and by the strength of their PATTERN. The refinement process produces regularities of greater generality and/or greater strength, which are desired because they apply to a larger fraction of the population and/or produce stronger, more unique predictions. The strength of PATTERN is measured by the chi-square test and the corresponding probability, and by several other statistical measures, such as Cramer coefficient. A threshold of acceptance can be set for each measure. Descriptions of contingency tables, various statistical parameters, and their efficient implementations are provided in Press *et al.*(1989) and in many books on statistics.

Theory is a collection of regularities. Forty-Niner tries to cover as much data as possible with regularities. It tries to come up with as few regularities as necessary; each as strong and as general as possible. The simplicity results from the order of search: try to cover all data for two variables with one regularity; if not possible then split the range into two subranges and cover each with a regularity. Continue recursively until all data are covered by regularities or until the dataset is too small to be split further.

Our definition of a regularity is compatible with other approaches. In first-order logic, every contingent formula (neither a tautology nor its negations) is a claim that excludes certain situations and therefore is a candidate for a regularity. A contingent formula is true in some possible worlds and false in others, describes a limitation in the set of all possibilities. Our definition fits the falsificationist view of science because each formula which claims that some possibilities never occur is falsified if they are shown to exist. It also satisfies the information measure approach. The less likely it is that the empirical distribution would have been randomly generated as the null hypothesis distribution, the more information it contains and the stronger a regularity it is.

4 Multisearch in 49er

Different elements of a regularity in the form (2) are determined by separate search components of 49er. All ranges are produced by the Partition-Data Search. Ranges are defined initially by an apriori decomposition of values of each attribute into subsets, later by the ranges refined in feedback with regularity refinement. All combinations of the attributes which occur in the pattern are selected by Select-Attributes Search, while the actual pattern of each regularity is determined by the Find-Regularity Search. Regularities in which patterns include more than two variables are obtained by the refinement search which tries to generalize two-dimensional regularities. The combination of all searches is illustrated in Figure 2.

4.1 Operations on Attributes

Forty-Niner.b performs three basic operations on each attribute: aggregation, slicing, and projection.

Aggregation. Aggregation combines values of an attribute into classes of abstraction. By reducing the number of values, we increase the relative density of data, allowing for efficient screening of large ranges of data by the fast computation of simple statistical regularities. Typically, Forty-Niner aggregates all values of an attribute into two classes: lower and higher. The system tries to ensure that the sets of records corresponding to both aggregates are of equal size, making the best possible approximation. However, other aggregation methods can also be used. For example, the data could be divided into more than two subsets, or the dividing point could be chosen by a different method.

Slicing. We call the next operation slicing, or data partitioning. Taking a slice of a data set using the value v_i of attribute A_i means selecting all elements of the data set that have value v_i. Slicing reduces the amount and narrows the range of data. If a regularity is weak when we consider all values of the attribute A_i, it is possible that stronger regularities exist in one or more slices of A_i. When slicing works on the results of aggregation, and v_i is an aggregate of values, all elements with a value of A_i in v_i are included into the slice. Section 6.1 shows how domain knowledge about attributes is used to produce slices.

Projecting. Projecting the attribute A_i is equivalent to ignoring this attribute; all records are included regardless of the value of A_i. In the multidimensional space W introduced in the

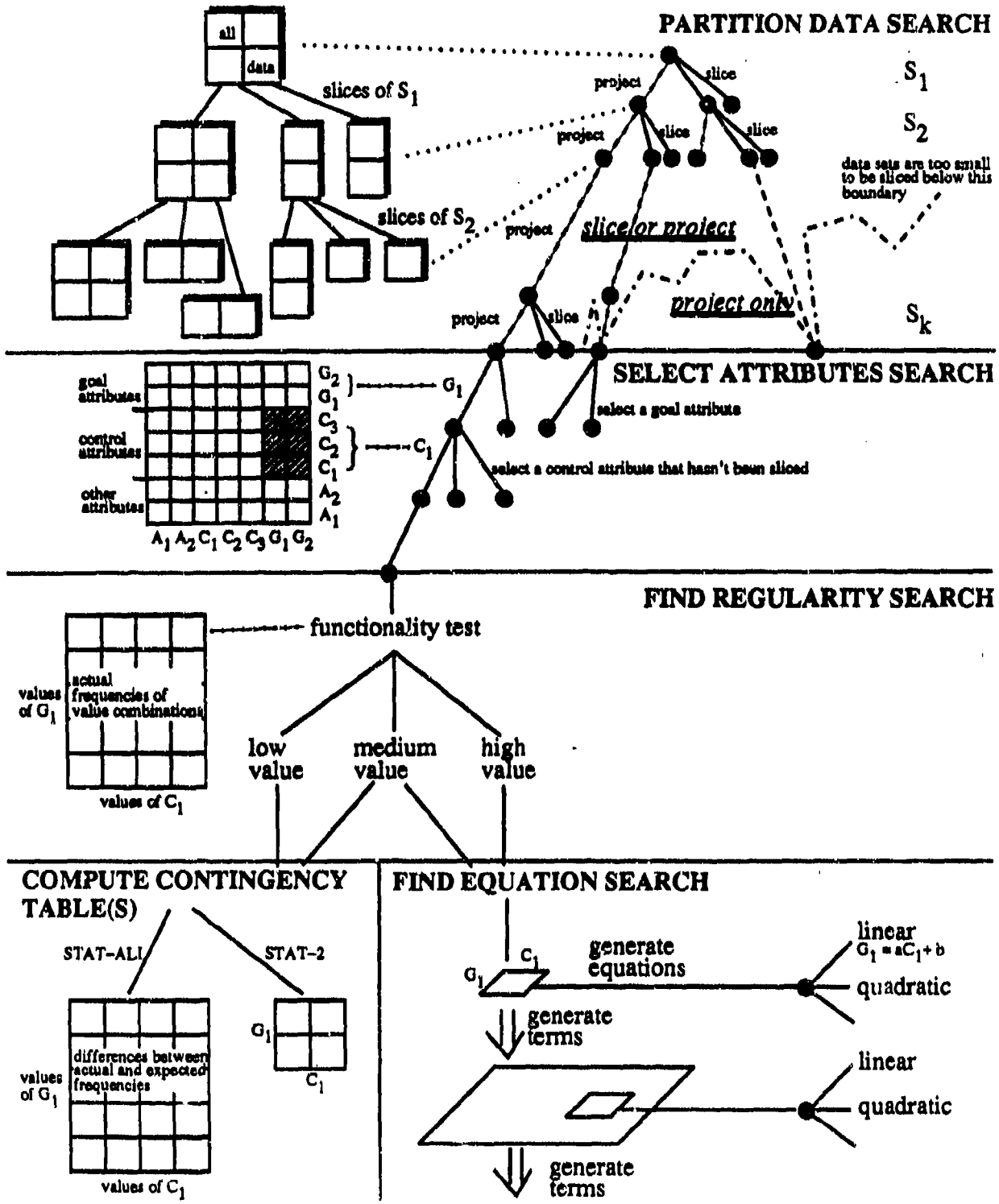

Figure 2: Overview of the search for two-dimensional regularities. The search, which proceeds from the top, is a combination of three searches: Partion-Data Search, Select-Attributes Search, and Find-Regularity Search. See section 4 for details of each search. To the left of the search tree are illustrations of some important data structures related to the search.

Section "What is a Regularity?", where each attribute represents one dimension, slicing and projecting reduce the number of dimensions. Slicing reduces the data set, while projecting preserves all the data. Projection and aggregation are useful in reducing the sparsity of data in W.

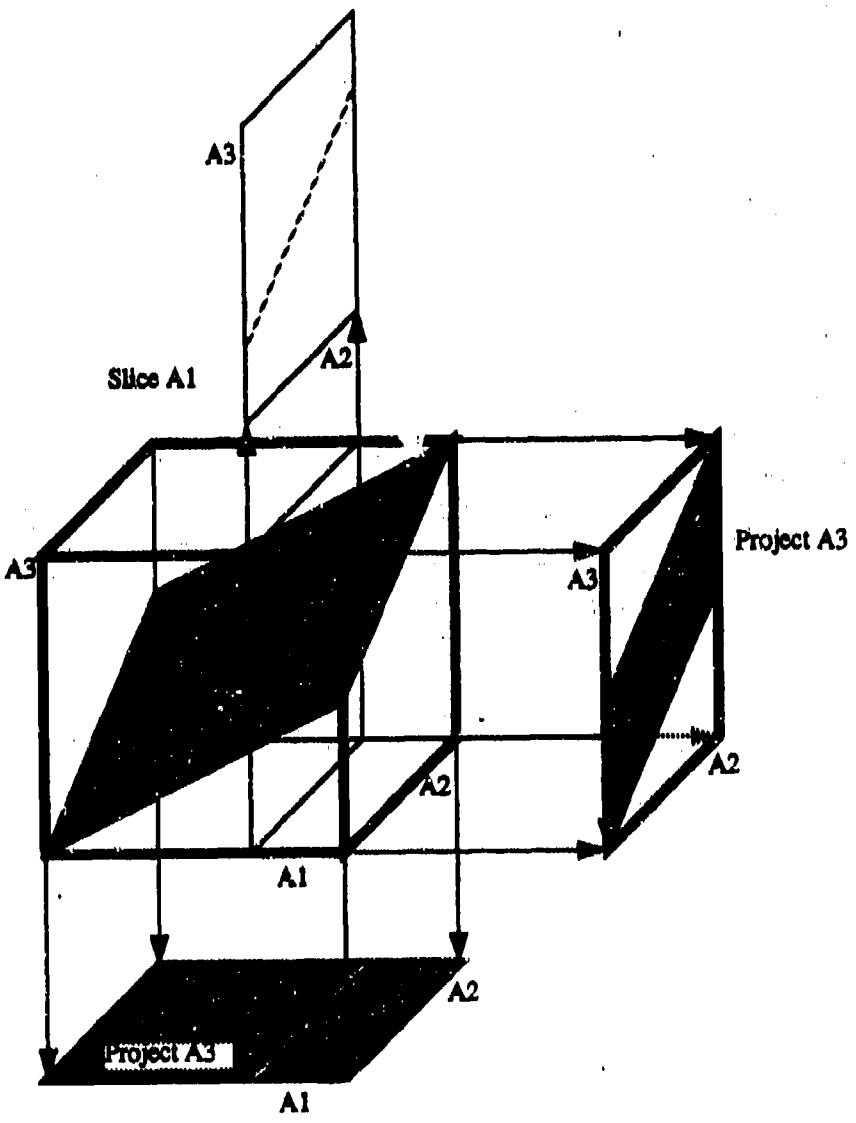

Figure 3: The partitioning search: slicing and projecting data

In Figure 3 we show examples of slicing and projecting in a three-dimensional space. The data in this space follow a strong regularity: $A_3 = (A_1 + A_2)/2$. If the space is sliced using a particular value of A_1, then a strong linear regularity is found between A_2 and A_3. However, if it is projected along A_1 instead, only a weak linear relationship can be detected. If the values of A_1 are first aggregated and then a slice is taken using one of the aggregates, then a moderately strong linear relationship can be found between A_2 and A_3. If the variable A_3

is used for projection, the regularity is virtually lost as a result.

4.2 Partition-Data Search

The Partition-Data search (upper part of Figure 2) produces subsets of the database, which are candidates for the RANGE of regularities and will be examined for patterns. 49er.b uses a simple step-wise partitioning mechanism, based on the values of one attribute at each step.

Aggregates of attribute values. Suppose that all the values of attribute A_i are aggregated into p disjoint classes or bins, $a_i^1 \ldots a_i^p$. Each class a_i^j defines a *slice* of a dataset D, including all records in D, for which a_i^j is the value of A_i. For the preliminary search, a practical number of slices per attribute is 2, but it could be any number. For simplicity, throughout this paper we will use $p = 2$, corresponding to two slices per attribute.

Attribute type determines aggregation. Different aggregates make sense depending on the relationships between attribute values. Here 49er.b distinguishes two types of attributes: *nominal* attributes, for which no ordering relationship between values is known, and *ordinal* attributes, the values of which are ordered linearly. To avoid spurious slices, 49er.b does not aggregate nominal attributes, using their original values if the number of values does not exceed p. Although nominal values can be aggregated, many alternatives are possible and most have little meaning. For ordinal attributes, the values are split into p categories, such that the corresponding subsets of records in the corresponding data set are as close as possible to equal. For $p = 2$, there is a single value a_i of A_i that defines two aggregates: $\{v|v \leq a_i\}$ and $\{v|v > a_i\}$. These sets contain "lower" and "higher" values of A_i.

Search control. Partitioning is controlled by the subset $S = \{S_1, \ldots, S_k\}$, $S \subseteq \{A_1, \ldots, A_N\}$, of attributes to be used in slicing, by the type of attribute, and by the minimum number m of records in a data set under which it will not be sliced. For each attribute in S, the data set is either projected (dotted lines), or sliced (solid lines in upper part of Figure 2) until the number of records is smaller than m, because the statistics for small sets are not reliable. The minimum slice size limits the depth of the partitioning search, as depicted by the boundary on the minimum number of records in the upper right part of Figure 2. Projecting an attribute is equivalent to ignoring its values. No records are excluded by the projection.

Search complexity. When 49er.b considers only two slices for each attribute in $S = \{S_1, \ldots, S_k\}$, plus the projection, the total number of possible ranges is 3^{k-1}. But the minimum slice size m limits data partition to approximately $\log_2 M/m$ steps. If $N = 30$, $M = 10^6$, and $m = 10^3$, then 3^{29} leaf nodes are reduced to only 3^0, by the factor 3^{23}. Since data are seldom sliced exactly into halves, the depth, at which branching stops, varies.

Summary of RANGE description. Range is described by a conjunction of conditions, each on the values of a single attribute. The number of conjuncts is limited by the minimum slice size. Formally:

```
Range-condition = Slice-condition & Range-condition  |  0
Slice-condition = Ai > ai  |  Ai <= ai
|Range| > m
```

4.3 Select-Attributes Search

The Select-Attributes Search (Figure 2) generates all combinations of control (independent) and goal (dependent) variables for each data subset generated by the Partition-Data Search. Then, at each leaf, the Find-Regularity Search for two-dimensional regularities is called.

Whether a variable is independent or dependent relies on the user goals and user control capabilities. Useful regularities hold between control variables, whose values can be set by user actions, and goal variables, those that the user wants to change, but whose values can be only controlled indirectly, throughout control variables. If the goal and control attributes cannot be decided, all attribute pairs are compared in an exhaustive search. The array which captures the knowledge of potentially useful pairs (C_i, G_j) of control and goal attributes (Figure 2; to the left of the Select-Attributes Search tree), controls node generation in Select-Attributes search.

If a regularity R has been detected for the attributes C and G in a data set D, typically an avalanche of regularities can be detected in the subsets of D for the same variables. Although all these regularities are worth considering because they can lead to refinements of R, this is done later by the refinement module, while the initial Select-Attribute search uses the following dynamic heuristic: "do not consider a pair attributes in a data subset if a regularity for those two attributes was already found in a superset". This heuristic works because the Partition-Data depth first search control never generates a subset of data before it considers all supersets of that subset. In fact, it never generates the same subset of data twice.

Search complexity. For K independent variables and L dependent variables, the number of pairs is $K \times L$. For the exhaustive search, when control and goal variables are not identified, the number of attribute pairs is $N(N-1)/2$, where N is the number of variables.

4.4 Find-Regularity Search

Given a range of data and two attributes, the Find-Regularity Search (two lower parts of Figure 2) looks for patterns that fit the right-hand side of a regularity (2). The search for each kind of regularity starts with an application test, followed by hypothesis construction and evaluation.

Domain knowledge determines regularity type. Application tests use domain knowledge about both attributes. Regularities in a particular type can make sense for some attributes only. For example, equations make sense only for numeric attributes which belong to the interval scale. When numbers do not mean more than names, an equation discovered for such a nominal attribute would be spurious, because it would vanish when the names or the ordering of names is changed. Attribute types belong to domain knowledge. We discussed how nominal and ordinal types are used to determine aggregates of values. If, in addition to ordering, the distance between each value and its immediate successor is the same, we get an *interval* attribute. All attribute types can be defined both on symbolic and numeric values, but since interval scale permits equation detection, the values must be converted to numbers, so that numeric techniques can be used.

The second concern is the number of values of an attribute. Although contingency tables are defined for any number of values, they become computationally costly and/or sparse if

the number of values is large. For this reason, the values are binned (aggregated) if their number exceeds a threshold (default of 10 in 49er.b). For the sake of equation generation, aggregation preserves the interval scale.

Application conditions for different types of regularities are summarized in Table 3. Additionally, the search for equations is triggered when the functionality test is passed, that is only when there is an approximate functional dependence between C_i and G_j.

Table 3: Dependence between type of both attributes and type of regularity.

	nominal	ordinal	interval	unbinned numeric
nominal	contingency-all	contingency-all	contingency-all	
ordinal	contingency-all	contingency-all contingency-2	contingency-all contingency-2	
interval	contingency-all	contingency-all contingency-2	contingency-all contingency-2 equation	equation
unbinned numeric			equation	equation

Functionality test. Empirical equations are efficient summaries of data, but the more distant the data are from a functional relationship, the less sense they make. Because the search for equations is expensive, 49er.b applies a functionality test, and looks for equations only when the test returns high value, for low values it only computes statistical contingency tables (Figure 2). The functionality test is based on the general definition of a function and the assumption of normal error. It is performed on the table of the actual frequencies of value combinations for both attributes.

Adding new patterns. As 'pattern' is a very broadly defined term, many other regularity types are possible, for instance, multifunctions (Piatetsky & Matheus 1991). The search for new types of regularities can be easily added to the Find-Regularity search, by specifying application condition, hypothesis construction mechanism, and evaluation criteria.

Hypothesis generation. All regularity types which survive the application tests are tried. The hypothesis construction mechanism creates the best instance for each type. This is done by direct computation (building a contingency table) or by yet another search (Equation Finder; its combination of two searches, generate terms and generate equations, is depicted in the lower right corner of Figure 2; for details see Zembowicz & Żytkow, 1991).

Hypothesis evaluation. When all elements of a hypothesis in the form (2) are generated, the evaluation is possible. Positive evaluation of regularity R justifies contributions made to R by all three searches. To establish the statistical significance or strength, for each pattern in the form of a contingency table 49er.b uses the χ^2-test and computes the probability Q that this value of χ^2 (or larger) could be the result of random fluctuation. A large value of the probability Q means that the regularity can be explained as a statistical fluctuation. All regularities with the probability Q larger than a user-controlled threshold are rejected. The threshold can be lowered if we want to review only the strong regularities. Evaluation of equations is based on a similar mechanism and discussed in detail in Zembowicz and Żytkow (1991).

Although the combination of searches depicted in Figure 2 exhaustively goes through the hypothesis space, the search is applied depth-first. The total number of hypotheses depends on search parameters and on a database. Table 17 shows that the maximum number of CONTINGENCY-all tables is 25,965, computed in an exhaustive search in a *DB* of 10,000 records, 10 attributes, 10 values per attribute, and a minimum size of a slice equal to 1,000. Thus, at the rate of 1 regularity/second, for more than 1000 records, 49er considers about 5 hypotheses per second. Typically however, the search is faster because the domain knowledge reduces search.

5 Types of Regularities

The Partition-Data search chooses a database subset, then the Select-Attributes search selects two attributes and invokes the search for regularities. Based on the domain knowledge, the list of applicable regularity types for those attributes is formed. The search for regularities finds all regularities from the applicable types and returns those that pass the evaluation criteria.

Currently the system considers regularities based on contingency tables and equations. Thanks to the open architecture of 49er, new types of regularities can be very easily added to the system.

In general, the search for each kind of regularity can be defined by specifying application criteria, detection mechanism, and evaluation criteria. Usually, a regularity makes sense only for attributes of some special type. For example, equations make sense only for numeric attributes, but, obviously, not for nominal. Application criteria provide requirements for attributes and the number of their values that must be satisfied to trigger the search for a given type of regularity. The hypothesis detection mechanism constructs the instance of the given regularity type. The detection could be a simple step (like building a contingency table) as well as another search (like Equation Finder). Due to the presence of noise in data and the degree of fit, each found regularity must be evaluated to establish its level of significance. Only regularities that pass those evaluation criteria will be further considered and reported.

5.1 CONTINGENCY-all Regularities

Let's imagine that we are interested in finding a regularity between days of leave (LEAVE) and success-rate at work (SUCCESS). If we want to check how values of the independent attribute LEAVE determine the values of the goal attribute SUCCESS, we can use a contingency table. Each entry in the contingency table A (see Table 4) is equal to the number of data records that have the corresponding values of both attributes (for example, there is 201 records with LEAVE from 11 to 20 days and SUCCESS= 100% at the same time). For a given value of the attribute LEAVE, the contingency table gives the empirical distribution of values of the attribute SUCCESS. For instance, if a person has 4 leave days, then the most probable values of SUCCESS are 67% and 100%

However, before any predictions based on a contingency table can be done, one must verify that indeed there is a dependence between considered attributes. If there is no correlation

Table 4: Days of leave vs success: actual counts

SUCCESS						
100%	342	201	290	315	117	127
67%	221	135	204	225	77	191
33%	95	35	67	66	33	103
0%	39	14	13	16	16	115
	0–10	11–20	21–30	31–40	41–50	51–60
						LEAVE (days)

between attributes, then the entries in the table are determined by individual distributions of values of each attribute. However, a scientist almost never knows *true distributions* of attribute values (this especially holds for databases). But if a given database is believed to be a representative sample of the whole population, then *empirical histograms* could be good estimation of those unknown distributions.

Let $h(x)$ be the histogram of attribute x,

$$h(x) = \{(x_1, n_{x_1}), (x_2, n_{x_2}), \ldots, (x_k, n_{x_k})\},$$

where $x_1, \ldots, x_k$ are values of attribute x while $n_{x_1} \ldots, n_{x_k}$ are counts of records with corresponding value of x. For the example in the Table 4,

$$\begin{aligned} h(\text{LEAVE}) &= \{(10, 697), (20, 385), (30, 574), (40, 622), (50, 243), (60, 536)\}, \\ h(\text{SUCCESS}) &= \{(0\%, 213), (33\%, 399), (67\%, 1053), (100\%, 1392)\}. \end{aligned}$$

If attributes x and y are independent, then the expected number of records with $x = x_i$ and $y = y_j$ is equal to

$$E_{ij} = \frac{n_{x_i} \cdot n_{y_j}}{N},$$

where N is the total number of records. This reflects the fact that the joint distribution of independent variables is an appropriate multiplication of distributions of each variable. The table E is usually called a table of expected counts, therefore contingency table A is sometimes called table of actual counts. The third table, δ, contains relative differences between actual and expected counts,

$$\delta_{ij} = \frac{A_{ij} - E_{ij}}{E_{ij}}.$$

Tables 5 and 6 show expected counts and relative differences for LEAVE vs SUCCESS.

We use the chi-square test to determine the significance of correlation between attributes based on the contingency table. The chi-square is defined as

$$\chi^2 = \sum_{i,j} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

The chi-square value is a global measure of the difference between expected and actual counts. However, it depends on the size of the contingency table and the number of records,

Table 5: Days of leave vs success: counts expected based on the independence hypothesis

SUCCESS						
100%	317.4	175.3	261.4	283.2	110.6	244.1
67%	240.1	132.6	197.7	214.3	83.7	184.6
33%	91.0	50.3	74.9	81.2	31.7	70.0
0%	48.6	26.8	40.0	43.3	16.9	37.3
	0–10	11–20	21–30	31–40	41–50	51–60

LEAVE (days)

Table 6: Days of leave vs success: table of differences

SUCCESS						
100%	0.08	0.15	0.11	0.11	0.06	−0.48
67%	−0.08	0.02	0.03	0.05	−0.08	0.03
33%	0.04	−0.30	−0.11	−0.19	0.04	0.47
0%	−0.20	−0.48	−0.67	−0.63	−0.06	2.08
	0–10	11–20	21–30	31–40	41–50	51–60

LEAVE (days)

therefore it cannot be directly used to verify the significance of the regularity. 49er estimates the probability Q that this obtained value of the chi-square (or smaller) could be the result of random fluctuation. Large value of the probability Q means that the regularity is statistically insignificant — such dependencies between involved attributes can be explained by, for example, noise present in data. The system uses a user-controlled threshold: all regularities with the probability Q larger than the threshold Q_{max} are rejected. The same evaluation method is used for all regularities based on contingency tables.

The probability Q measures the statistical *significance* of a contingency table. It tells only what are the chances that the corresponding value of chi-square could be generated by a random fluctuation. There are other measures that try to estimate the *strength* of the correlation between given attributes. The strongest, or ideal, correlation is when for each value of one attribute there is at most one corresponding value of the other attribute, and vice versa. In other words, in each row and in each column of the contingency table for those attributes there is at most one non-zero entry (see Table 7). Chi-square should not be used to judge the strength because it highly depends on the number of records and the size of the contingency table. 49er computes two other measures: Cramer's V and the contingency coefficient C.

For a given $N_{row} \times N_{col}$ contingency table, Cramer's V is defined as

$$V = \sqrt{\frac{\chi^2}{N \min(N_{row} - 1, N_{col} - 1)}},$$

where N is the number of records. For ideal correlation, χ^2 is equal to $N \min(N_{row} - 1, N_{col} - 1)$. Thus we see that for a perfect correlation Cramer's $V = 1$. On the other extreme, when actual counts are exactly equal to expected, then $\chi^2 = 0$ and $V = 0$. The larger Cramer's V

Table 7: Example of a contingency table for ideal correlation: for each value of the attribute x there is at most one value of the attribute y, and vice versa.

Attribute y				
high	27	0	0	0
moderate	0	0	0	44
low	0	0	76	0
	a	b	c	d

Attribute x

is, the stronger correlation is. Therefore this measure can be used to compare contingency tables.

The other measure computed by 49er is the contingency coefficient C,

$$C = \sqrt{\frac{\chi^2}{\chi^2 + N}}.$$

Its value is 0 when there is no correlation, but its upper limit depends on the size of the table. Therefore it can be used only to compare contingency tables of the same size.

5.2 CONTINGENCY-2 Regularities

Sometimes a CONTINGENCY-all regularity is too detailed to be easily interpreted, because the corresponding contingency table could be quite large (see Table 4). However, we can introduce four new concepts by grouping values of each attribute into two subsets: "lower" and "upper". For the example discussed in the section 5.1, the four new concepts are: SMALL-SUCCESS, BIG-SUCCESS, SHORT-LEAVE, and LONG-LEAVE. Tables of actual and expected counts formed for these four concepts (see Table 8) suggest the regularity: short leave correlates with big success. The CONTINGENCY-2 table is evaluated as a 2 by 2 contingency table.

Table 8: CONTINGENCY-2 for LEAVE vs SUCCESS: actual counts

BIG-SUCCESS	833	559
SMALL-SUCCESS	823	842
	SHORT-LEAVE	LONG-LEAVE

Table 9: CONTINGENCY-2 for LEAVE vs SUCCESS: expected counts

BIG-SUCCESS	754.1	637.9
SMALL-SUCCESS	901.4	763.1
	SHORT-LEAVE	LONG-LEAVE

Table 10: CONTINGENCY-2 for LEAVE vs SUCCESS: differences

BIG-SUCCESS	0.105	−0.124
SMALL-SUCCESS	−0.088	0.103
	SHORT-LEAVE	LONG-LEAVE

The only problem is how to define border line between concepts (is 30 days long or short leave?). If there is no hint in the domain knowledge that could suggest definition of concepts, the system attempts to introduce concepts itself. The initial partitioning of LEAVE to SHORT and LONG is based on a simple rule: number of records (people) with short leave should be approximately equal to that with long leave. If the resulting regularity for concepts is quite significant (statistically strong), then one could try to refine the definition of concepts to strengthen the regularity by varying borders between concepts. However, this refinement could be quite costly and thus it is not done for all CONTINGENCY-2 regularities (see section 7.3). For the above example, 49er defined BIG-SUCCESS as 100%, SMALL-SUCCESS as 0%, 33%, or 67%, SHORT-LEAVE as 0–30 days, and LONG-LEAVE as 31–60 days.

5.3 Equations

49er.b incorporates Equation Finder (EF; Zembowicz & Żytkow, 1991), another machine discovery system. Equation Finder can detect a broad range of equations useful in different domains, and can be easily expanded by addition of new variable transformations. Previous systems, such as BACON or ABACUS, disregarded or oversimplified the problems of error analysis and error propagation, leading to paradoxical results and impeding the true world applications. Our system treats experimental error in a systematic and statistically sound manner. It propagates error to the transformed variables and assigns error to parameters in equations. It uses errors in weighted least squares fitting, in the evaluation of equations, including their acceptance, rejection and ranking, and uses parameter error to eliminate spurious parameters. The system detects equivalent terms (variables) and equations, and it removes the repetitions.

Input to EF consists of N numeric data points (x_i, y_i, σ_i), $i = 1, \ldots, N$, where x_i are values of the independent variable x, y_i are values of the dependent variable y, σ_i represents the uncertainty of y_i (scientists call it *error*, for statisticians it is *deviation*, while the term *noise* is often used in AI). Output is a list of acceptable models.

Model Fitting. Model Fitter, a component of EF, uses *chi-square fitting*, known also as *weighted least-squares* (Eadie *et al.* 1971, Press *et al.* 1989) to fit given numeric data points (x_i, y_i, σ_i) to a finite number of models. Model is a function template $y = f(x, a_1, \ldots, a_q)$ (for example, $y = a_1 + a_2 x + a_3 x^2$) whose parameters' values $a_1, \ldots, a_q$ are determined by the requirement that the value of χ^2,

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i, a_1, \ldots, a_q)}{\sigma_i} \right)^2 \tag{3}$$

is minimal. The value of χ^2 is the sum of squares of deviations of data points (x_i, y_i) from the model, weighted by errors σ_i, so that measurements which are more precise carry greater weight. At the minimum of χ^2, its derivatives with respect to the a_j all vanish,

$$\sum_{i=1}^{N} \frac{y_i - f(x_i, a_1, \ldots, a_q)}{\sigma_i^2} \cdot \frac{\partial f(x_i, a_1, \ldots, a_j, \ldots, a_q)}{\partial a_j} = 0, \tag{4}$$

for $j = 1, \ldots, q$. In general, the set (2) of equations is non-linear and not easy to solve. For a polynomial model, however, the set (4) can be solved by algebraic or matrix operations,

producing efficiently a unique solution. Our Model Fitter can consider polynomials of any degree. In addition to the original variables x and y Model Fitter can work on data expressed in any transformed variables: $x' = x'(x)$ and $y' = y'(x, y)$.

Parameter Error. Standard deviations of parameters $a_1, \ldots, a_q$ at the values that minimize χ^2 are calculated according to the formula for error propagation (Eadie *et al.* 1971):

$$\sigma_{a_j}^2 = \sum_{i=1}^{N} \left(\frac{\partial a_j}{\partial y_i} \right)^2 \cdot \sigma_i^2, \qquad j = 1, \ldots, q \tag{5}$$

(parameter values $a_j, j = 1, \ldots, q$ are solutions of equations (4), therefore they are functions of x_i, y_i, σ_i).

Removing vanishing parameters. If the absolute value of a fitted parameters a_j is smaller than the corresponding error σ_{a_j}, then zero could be also a good value for a_j. EF sets this parameter to zero. The new simplified equation has a similar goodness of fit (Zembowicz & Żytkow 1991).

Fit Evaluation. For the best fit to each model, EF calculates the value of χ^2 according to Equation (3), and assigns the probability $Q = Q(\chi^2, N - q)$ ($N - q$ is the number of degrees of freedom in the data left after the fit) that this χ^2 value has not been reached by chance. A small value of Q means that the discrepancy between the data (x_i, y_i, σ_i) and the model $y = f(x, a_1, \ldots, a_q)$ is unlikely to be a chance fluctuation. The *threshold of acceptance* is defined using the probability Q: all models for which Q is smaller than some minimum value Q_{min}, are rejected. The best models are those with the largest values of Q.

Figure 4 demonstrates the generation of new variables and equations. It depicts the backtrace of actions needed to generate the equation $y = e^{(a+bx^2+cx^4)/x}$; this equation can be generated at depth two in the Generate New Variables search (the original variables x and y are depth zero).

6 Use of Domain Knowledge

49er uses existing domain knowledge to formulate goals for database exploration, to optimize the search for regularities, and to avoid considering spurious regularities. The system recognizes the following kinds of knowledge:

- attribute dependency information: which attributes are goal or dependent attributes, which are control or independent; this information is used to determine exploration goals;
- attribute type: used by the search for regularities to consider only those regularity types that make sense for considered attributes (see section 6.2) and by the partitioning search to produce slices (subsets) of the database (see Section 6.1);

In addition to the information listed above, 49er also uses symbolic names and description of attributes, names of defined groups and concepts, etc, to ease the communication with the user.

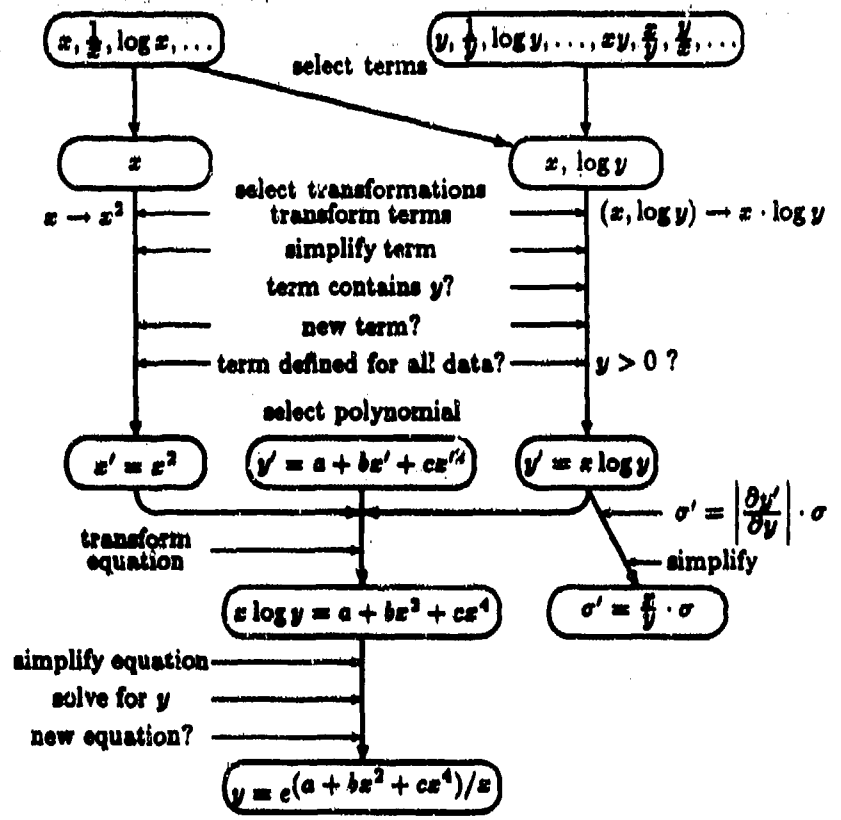

Figure 4: Generation of new variables and equations: the backtrace of all actions responsible for the generation of equation $y = e^{(a+bx^2+cx^4)/x}$.

Now let us present attribute types that 49er currently recognizes.

An attribute of type nominal has simply symbolic values. For convenience, 49er assumes that the set of possible symbolic values is finite. If each symbolic value has a unique predecessor and a unique successor (with the exception of the first and the last value, respectively), then an attribute of such type is said to be ordinal. If, in addition to ordering, there is a measure that defines the distance between any two values and the distance between a value and its successor is always the same, the attribute is called an interval one. All these three types (nominal, ordinal, interval) have symbolic values. But there could be also number-valued attributes. Numbers could be treated as values of the type ordinal, but the set of possible values of a numeric attribute is infinite. 49er recognizes two numeric types: real or 'plain' numeric (value is any real number) and 'binned' numeric. For the latter one, numbers are grouped into bins, also called intervals. Thus, possibly, the set of values of a binned numeric attribute becomes finite. If the length of each interval (bin) is the same, then the type binned numeric becomes compatible with the type interval.

6.1 Dependence Between Attribute Type and Slicing Method

The way an attribute is used to produce slices (subsets) of the database depends on the type of this attribute and and the set of its values. The partitioning search uses here two

user-controlled thresholds: the maximum number p of slices based on one attribute and the minimum size (number of records) of a database subset.

If the number of values of a nominal attribute is not larger than the threshold p, each produced slice will correspond to one attribute value, that is, all records in a slice will have the same value of the attribute used to split the database. If slices cannot be produced this way, then the actual set of data is split into two subsets such that both have approximately the same number of records. The determination of the subsets is the same as the definition of concepts defined by a CONTINGENCY-2 regularity. The last method works only for ordered attribute values, that is, for ordinal, interval, ratio, numeric, and binned numeric. It means that if an attribute of type nominal has too many values, it cannot be used to slice the data to subsets.

6.2 Dependence Between Attribute Type and Regularity Type

It is clear that each regularity discussed above requires attributes to be of some particular type — for other types the regularity does not have any sense (that is, it is impossible to interpret such regularity). For example, equation for nominal attributes would be a very strange regularity. Therefore this section discusses the dependence between the type of attribute and the type of regularity. The graph presented in Table 3 summarizes the dependence between the attribute type and the regularity type.

CONTINGENCY-all regularities can be considered for all symbolic types: nominal, ordinal, interval, and ratio. In addition, if an attribute of numeric type was binned (grouped into intervals), then CONTINGENCY-all regularity will be analyzed also for this attribute.

Initial definition of concepts used in CONTINGENCY-2 is based on the assumption that there is an order among the values of considered attribute. Therefore nominal attributes cannot be treated in the same way as ordinal. Thus CONTINGENCY-2 regularity is considered only for ordinal, interval, ratio, and numeric types.

Equations can be found only for interval, ratio and numeric attributes, either plain (type real) or binned into groups of equal length.

We now see that binning values of a numeric attribute is very welcomed because 49er can then uncover a regularity between symbolic attribute and numeric one. In addition, the search for equations is expensive (compared to the analysis of contingency tables). Binning data reduces time needed to find an equation, because, usually, the number of values is significantly reduced, too. If an equation is discovered, 49er can refine it by taking the original, unbinned data (see Section 7.4).

7 Regularity and Concept Refinement

The regularities found by the initial search can be further improved by another search. The goal of this search is to refine already discovered regularities and concepts, to make them stronger. 49er.b uses several techniques to expand the range of regularities and/or to increase the strength of their patterns. Stronger and more general regularities lead to more unique predictions and better decisions. They also lead to concepts that better fit the domain. All improvements can be considered in the same framework of searches as the initial discovery

of regularities. All the resultant regularities have the same form (2), no matter by what combination of refinement techniques they have been reached.

7.1 Refinement of Range

Initial selection of range is done to some extent *a priori*, that is after binning data based on the computed histograms for the sliced attributes, but before any regularities have been found. But after a regularity is found, the range can be improved by additional search. By the discussed properties of control of the Partition-Data Search, for each regularity $D \rightarrow REG(C, G)$, no regularity for C and G has been found in the supersets of D, or the search would stop there. But only a limited number of supersets has been considered and small increases of the range are still possible.

The improvements are done by the following combination of three searches:

```
Until pattern strength is no longer improved:
  For each attribute in the description of D:
    Vary slightly the range D,
  For each new range:
    Use the same attributes C and G
      Find pattern, compute the strength
  Select new D for which pattern is strongest
```

The range is varied by changing one of the conditions at a time. For an ordered attribute S_i, if a condition has been $S_i > a_j$ and if the immediate neighbors of a_j are a_{j-1} and a_{j+1}, then the new conditions are $S_i > a_{j-1}$ and $A_i > a_{j+1}$; each new subset should exceed the minimum slice size. For a nominal attribute, the values are aggregated in all combinations, which does not cause a very large number of combinations, because only those attributes with a very small number of values have been used in Partition-Data Search. The search is controlled by hill climbing.

7.2 Adding New Dimensions to Patterns

A 2D regularity can be a projection of a finer regularity in more dimensions. It can be also a projection of a regularity in one slice of attribute A_i and random distribution in another slice. We will now discuss how 49er deals with both problems. In the former case, the improved regularity has the same range but a stronger pattern, while in the latter, the range is smaller but the pattern becomes much stronger.

```
For a yet unsliced attribute A:
  Generate slices
  For each slice:
    Select the same attributes G, C1,...,Ck
    Find 2 dimensional regularity
  If there is a stronger regularity in each
      slice (compared to original regularity)
    then
```

```
    Select the attributes G, C1,...,Ck, A
    Find k+1 dimensional regularity
  else
    combine slices for which regularities
        are stronger than the original
        regularity
    Select the attributes G, C1,...,Ck, A
    Find k+1 dimensional regularity
```

This search mechanism is very similar to BACON and FAHRENHEIT. Both multidimensional equations and multidimensional contingency tables can be generated this way. Often, their pattern is much stronger than the pattern in a less-dimensional projections.

This refinement method works for all types of regularities, and can produce regularities in all types. For instance, if a regularity is first uncovered as a 2 dimensional CONTINGENCY-all table, it can be then refined to a 3 dimensional equation.

7.3 Refinement of CONTINGENCY-2 Regularities and Concepts

CONTINGENCY-2 regularities are contingency tables based on attribute values aggregated into two disjoint and complementary subsets of "lower" and "upper" values for each attribute. They are an important tool for summarizing regularities, very similar to linear correlations, because they are easy to interpret, to use for making predictions or decisions, and take little space even if generalized to many dimensions. The initial aggregation is done *a priori*, taking into account the histogram of each attribute, but not regularity between attributes. This way an initial approximation of a stronger regularity can be captured. Then, by the changes of the concept definitions for the "lower" and "upper" values, CONTINGENCY-2 can be strengthened. New concepts of "lower" and "upper" values obtained as a result of refinement can be more significant.

The refinement of CONTINGENCY-2 uses the same change-range operator as Refine-Range, and the same hill-climbing search control, which stops when strength can be improved no more. For ordinal attributes the resulting algorithm is fast ($O(k^2 \times n)$, where n is the number of values of the attribute, k is the number of attributes).

Tables 11 and 13 show an example of CONTINGENCY-2 refinement for a regularity found in a database of 3252 records. The success of recruitment depends strongly on the length of recruiter's leave in a given year (or it can be the other way around: the leave can depend on success). The dependence turns out to be particularly strong when leaves are divided into longer than 50 days and those not longer than 50 days. The refined regularity gives special meaning to a short leave (50 days or less), that is one which allows the recruiter to be successful and a long leave (more than 50 days) which correlates with significantly lower performance. The priori split at 30 days also categorizes leaves into short and long, but is not as significant.

Table 11: Original CONTINGENCY-2 regularity.
$\chi^2 = 33.11$, $Q = 8.7 \cdot 10^{-9}$

SUCCESS

> 67%	0.105	−0.124
≤ 67%	−0.088	0.103
	≤ 30	> 30

LEAVE (days)

Table 12: CONTINGENCY-all showing initial definition of "lower" and "upper" values.

SUCCESS

100%	0.08	0.15	0.11	0.11	0.06	−0.48
67%	−0.08	0.02	0.03	0.05	−0.08	0.03
33%	0.04	−0.30	−0.11	−0.19	0.04	0.47
0%	−0.20	−0.48	−0.67	−0.63	−0.06	2.08
	0–10	11–20	21–30	31–40	41–50	51–60

LEAVE (days)

Table 13: CONTINGENCY-2 regularity after refinement.
$\chi^2 = 173.13$, $Q = 1.5 \cdot 10^{-39}$

SUCCESS

> 33%	0.055	−0.258
≤ 33%	−0.219	1.032
	≤ 50	> 50

LEAVE (days)

Table 14: CONTINGENCY-all table showing refined "lower" and "upper" values.

SUCCESS

100%	0.08	0.15	0.11	0.11	0.06	−0.48
67%	−0.08	0.02	0.03	0.05	−0.08	0.03
33%	0.04	−0.30	−0.11	−0.19	0.04	0.47
0%	−0.20	−0.48	−0.67	−0.63	−0.06	2.08
	0–10	11–20	21–30	31–40	41–50	51–60

LEAVE (days)

7.4 Refinement of Equations

Refining an equation E means finding another equation that describes the same range of records, for the same attributes, better than the original one. An equation can be improved in two ways. First, if the values of either attribute were grouped into bins, then Equation Finder can be run on original, unbinned data. Since data aggregation increases error, the new equation will fit data more precisely. Second, Equation Finder can be run to the larger depth of search: the maximum degree of a polynomial and/or the maximum transformation level can be increased (Zembowicz & Żytkow 1991). Transformations can be also expanded in breadth, by considering additional primitive transformations.

7.5 Automated Use of Refinement

49er.b is typically used according to the following strategy:

Strategy 1 *Run the basic search, returning all regularities which exceed a permissive threshold of acceptance. Select a subset of regularities and refinement methods and apply refinement.*

Another, fully automated strategy could be also used:

Strategy 2 *Run the basic search returning only those regularities which exceed a high threshold of acceptance. Have the system select and apply all refinements automatically.*

Both strategies can be aided by a good guess of a threshold. If the threshold is low, a very large number of regularities can be generated. If the threshold is high, few regularities will be found. Strategy 2 requires a good guess of the threshold and good selection criteria for the regularities to be refined. If we do not have confidence in our formal criteria, it is better to use "manual" selection.

A hard problem for automated refinement is to set the tradeoff between the increase of range and of regularities and the tradeoff between improved strength and increased storage. Refinement is computationally expensive. Even if it may take a second per regularity in simple cases, this figure must be multiplied by the number of regularities. The branching and depth of most refinement methods depend on the number of attributes, which we attempt to incorporate in a given regularity.

8 Tests on Real and Artificial Data

49er's search is simple and modular, so that system analysis can confirm that all generated regularities fit the form given in (2). The computations leading to the scope and strength of individual regularities can be also justified by the analysis of search mechanisms and statistical methods of evaluation.

8.1 Application to real databases

New and challenging problems arise when we want to evaluate 49er's findings on real data. When applied to a megabyte size database, 49er typically discovers hundreds of regularities which are statistically very significant (a 200-600kB output file is typical). Most of them take the form of contingency tables, but functional regularities are also detected. Many regularities are subsequently refined to reach the strength of many orders of magnitude higher. Whether they can be further refined and whether some regularities have been missed is an open issue, partly because we do not have a clear definition of optimality, and partly because it is difficult to have an alternative source of answers to discovery problems for most of real-world databases. We tested optimality by a hide-and-seek method described later in this section, applying 49er.b on artificially generated databases with known regularities hidden in noise.

Table 15: One of results obtained by Chipman *et al.*: correlations of ability/attitude with career orientation. In italics the correlations for which 49er found counterparts. Career orientation attributes: PSCI: Physicist/Chemist & Engineer; BIO: Biologist & Doctor; BEX: Business Executive; LAW: Lawyer; J/W: Journalist/Writer. Ability/Attitude attributes: VSAT and QSAT: verbal and quantitative tests for high school students; HSmth: high school math background; HStchr: student's view of high school teachers; P/T: prefer work with people than things; Money: interest in income and money; Ideas: importance of working with ideas; MathA/C: math anxiety/confidence; MathAv: math average; CompA/C: computer anxiety/confidence.

	PSCI	BIO	BEX	LAW	J/W
VSAT	-0.08	-0.07	-0.09	0.04	*0.18*
QSAT	0.11	0.03	0.07	-0.03	-0.12
HSmath	*0.26*	*0.21*	0.14	0.02	*-0.21*
HStchr	*0.21*	*0.16*	0.09	-0.03	-0.16
P/T	0.31	*0.14*	0.12	-0.05	-0.12
Money	0.17	0.17	*0.44*	*0.23*	*-0.16*
Ideas	-0.17	-0.12	*-0.19*	-0.03	*0.39*
MathA/C	*0.45*	*0.31*	*0.14*	-0.07	*-0.32*
MathAv	*0.30*	0.18	0.06	-0.09	*-0.25*
CompA/C	*0.24*	*0.23*	0.07	0.03	-0.08

Table 16: 49er's results for the same data. Each entry is the confidence level (probability) Q of CONTINGENCY-2 regularity. Only regularities with $Q < 10^{-5}$ are shown.

	PSCI	BIO	BEX	LAW	J/W
VSAT					10^{-7}
QSAT					
HSmath	10^{-10}	10^{-7}			10^{-9}
HStchr	10^{-6}	10^{-6}			
P/T	10^{-20}	10^{-6}			
Money			10^{-23}	10^{-6}	10^{-6}
Ideas			10^{-8}		10^{-36}
MathA/C	10^{-33}	10^{-12}	10^{-6}		10^{-16}
MathAv	10^{-16}				10^{-14}
CompA/C	10^{-10}	10^{-6}			

8.2 Comparison with human findings

For most databases we do not know the "correct" answers which can be compared with the findings of 49er. A database which has been studied extensively by human researchers gives a good opportunity for comparisons. It is difficult to get human generated results, however, especially for large databases. Fortunately, we were able to test 49er on a database that was extensively analyzed by scientists. We set the goal to reproduce their results by 49er. In the next stage we will try whether 49er can do better on the same data.

Chipman, Krantz, and Silver (1990) analyzed mathematics anxiety among freshmen stu-

dents in the Bernard Women's College. The original data are answers to a questionnaire given to incoming students. 1366 records of about 125 attributes each (350kB of data) correspond to 1366 students answering 125 questions. In the first stage, Chipman *et al.* focused on combining individual questions, which are similar enough so that the answers can be totaled into compound attributes. Then they analyzed dependencies between compound attributes. We have run 49er following the same strategy. First, we run 49er on the same groups of original questions as considered by Chipman *et al.* reaching similar conclusions. 49er's results confirm that the questionnaire is well design to study math anxiety and allow us to select the same top five questions to define the compound attribute of math anxiety. Our results confirm that other forms of anxiety are weakly defined. In the process, 49er discovered many regularities, which can be the starting point for further refinement process, but our task was to parallel the reported results. Typically, it took 5 to 12 minutes to obtain regularities for one group of questions, about 10 seconds per one useful regularity.

In the second phase, 49er was executed on the compound attributes created by Chipman *et al.* The results were again very similar. A sample comparison is presented in tables 15 and 16. Chipman *et al.* use linear correlations in their analysis, while 49er uses CONTINGENCY-2 and CONTINGENCY-all contingency tables; nevertheless the results correspond very closely (large absolute value of the correlation coefficient corresponds to small value of the probability Q, and *vice versa*). In Table 15 the results in italics are those for which 49er found counterparts. 49er haven't discovered some weak correlations because the threshold of acceptance was set at $Q = 10^{-5}$.

We could draw several conclusions from our test. First, the applications of 49er which duplicated the results in Chipman's report were simple and schematic: "For a set of independent attributes, and a set of dependent attributes find regularities in the whole dataset." Slicing was applied only few times, on a single attribute. Those applications did not utilize the strength of 49er, which lies in fast search over large number of data subsets, and in detection of different types of regularities, depending on the data. 49er needed about 10 minutes to conduct each search, spending several hours on all tests altogether. In fact we applied 49er each time on a deeper search, obtaining many regularities, which can be starting point to the regularity refinement process and human analysis. The total output from 49er contains more than one and half megabyte of results. We were surprised that the our mechanism for knowledge representation applied so well to the research reported by Chipman *et al.* A long sequence of paragraphs in their report could be directly translated into search problems for 49er and all ten of their tables have counterparts in 49er's findings.

8.3 What can be Discovered in Purely Random Data?

Statistical fluctuations, if sufficiently improbable, are reported as regularities. In fact, for each contingency table, 49er.b computes the probability Q that the corresponding (or larger) value of chi-square could be the result of a random fluctuation. For each probability value q, if a sufficiently high number of hypotheses is considered during the search, a pattern with $Q < q$ can be present in randomly generated data. If 49er.b considers 10,000 hypotheses for purely random data, we can expect that approximately one of them has a probability Q of no more than 0.0001. Such a regularity considered in isolation may seem significant, but it is *only* a random fluctuation. This is the nature of probability: we must accept that some

statistical fluctuations can be mistaken for regularities.

This observation suggests the following test: run the system on random data, compute the statistics of regularity probabilities, and compare with theoretical predictions. About 90% of regularities should have $Q \geq 0.1$, about 9% should have Q in the range $0.01 \leq Q < 0.1$, and so forth. We have created 25 databases containing data generated randomly from uniform distribution. Table 17 shows test results for several databases. The second task is to advice the user on the acceptance threshold at which s/he can be reasonably sure that the regularities are real.

Table 17: Statistical fluctuations interpreted as regularities. All databases generated randomly from uniform distributions of values. 49er.b was set to report all regularities irrespective of their significance. Results are for CONTINGENCY-2 contingency tables; distribution of Q for CONTINGENCY-all regularities is very similar. (CPU time is reported for a DECstation 5000 running Allegro Common Lisp.)

Test number	9	13	18	24	25	
Number of records	10000	1000	10000	1000	1000	
Number of attributes	5	10	10	10	10	
Number of values	10	10	10	4	2	
Minimum slice size	200	100	1000	100	100	
Number of regularities	270	25797	25965	25944	25776	
Run time (CPU hours)	0:03	1:33	5:53	0:19	0:12	
Range of Q	Actual % of regularities					Expected
$1 \geq Q > 0.1$	92.96	90.02	90.36	89.81	89.52	90.000
$0.1 \geq Q > 0.01$	5.56	8.97	8.65	9.20	9.46	9.000
$0.01 \geq Q > 0.001$	1.48	0.91	0.89	0.88	0.95	0.900
$0.001 \geq Q > 0.0001$	0.00	0.09	0.08	0.10	0.06	0.090
$0.0001 \geq Q > 0.00001$	0.00	0.00	0.01	0.01	0.00	0.009

Let us generate a random database with n attributes $A_1, \ldots, A_n$. For each attributes $A_i, i = 1, \ldots, n$ the values of A_i in all N records are randomly generated numbers. In other words, such a database contains pure noise. We have created 25 such databases, varying the number of records, number of attributes, and the number of values per attribute. To get the full statistics, we have set 49er.b to report all regularities irrespective of their significance. Then, for each database, we have computed the histogram of all probabilities in probability Q, to verify whether Q has the distribution based on its probabilistic interpretation. Table 17 shows sample results for databases of different size. Results contained in the table show CONTINGENCY-2 regularities. Distribution of Q for CONTINGENCY-all regularities is very similar.

Table 17 provides us also with CPU time measurements and the number of hypotheses that are generated in sample databases. CPU time is reported for a DECstation 5000 running Allegro Common Lisp. 49er analyses some 8 regularities (CONTINGENCY-2 and CONTINGENCY-all) per second, on average, in a database of 1000 records, and about 2 per second in database of 10,000 records. The maximum number of hypotheses based on contingency tables that can be considered for 10 attributes compared to each other, and for three layers of slicing is 2×25965.

Should we mistrust regularities discovered by 49er, treating them as possible fluctuations? Not if we decide to accept only the regularities sufficiently stronger than possible fluctuations. If we know the amount of hypotheses to be considered by a particular search, we can estimate the strength of the strongest random fluctuations. Then, if we set the threshold on the acceptable probability by several orders of magnitude below that number, we can be sure that the regularities are real. An appropriate threshold is known as Bonferroni adjustment. For instance, if we set the probability thresholds at $Q = 10^{-7}$ for the search comparable to tests 13, 18, 24, and 25 in Table Table 17, the probability that a random regularity will pass the threshold in one search is 0.01. It means that among hundreds of regularities detected in a single search, there is 0.01 chance that one of them is a result of random noise.

Few additional data in Table 17 are worth mentioning. 49er was forced to report all contingency tables within depth of slicing set at about three, reporting some 25,000 regularities in individual runs, and spending from 0.027 to 0.8 second per regularity, depending on the number of records, and values per attribute.

Table 18: Test: discovery of hidden regularities.

Attributes	Regularities		
	Type	Significance	
A_0, A_2	contingency-all	$\chi^2 = 145.9$	$Q = 3.8 \cdot 10^{-30}$
A_0, A_4	contingency-all	$\chi^2 = 227.7$	$Q \approx 0$
A_1, A_2	contingency-2	$\chi^2 = 181.7$	$Q = 2.1 \cdot 10^{-41}$
	contingency-all	$\chi^2 = 635.2$	$Q \approx 0$
A_3, A_4	contingency-2	$\chi^2 = 236.6$	$Q \approx 0$
	contingency-all	$\chi^2 = 791.5$	$Q \approx 0$

Table 19: CONTINGENCY-all regularity between A_3 and A_4 for the whole range of data $\chi^2 = 791.5$, $Q \approx 0$

Attribute A_4

6	10	13	12	10	8	11	18
5	10	15	6	11	65	80	66
4	1	11	82	78	15	11	5
3	15	98	16	13	11	12	8
2	45	16	8	12	16	9	11
1	43	11	9	7	10	12	6
0	8	13	13	10	12	12	6
	1	2	3	4	5	6	7

Attribute A_3

8.4 Hide-and-seek Testing

Different regularities can be hidden in data in various combinations with noise, for the purpose of testing. As an example of gradual discovery, let us consider a 5-attribute database

with the following regularities. Attribute A_0 had values 0 and 1 generated randomly from uniform distribution. For $A_0 = 0$, the values of all other attributes were randomly generated. For $A_0 = 1$, the following relations were used:

$$A_2 = a + bA_1, \qquad A_4 = c + d \log A_3. \tag{6}$$

The values of A_1 and A_3 (ranging from 1 to 7) were generated randomly from uniform distribution; a, b, c, d are numerical constants; Gaussian error was added to A_2 and A_4.

Table 18 summarises regularities that hold for all data, found by 49er.b in the initial phase of search. Note that although strong regularities were discovered for all values of A_0, no equation has been found. Table 19 shows the CONTINGENCY-all regularity for attributes A_3 and A_4. Notice the effect of noise: all cells have non-zero entries because of noise, but the regularity is still very significant ($\chi^2 = 791.5$, $Q \approx 0$). The refinement search, which seeks to add new dimensions to the pattern, decomposed this regularity into pure noise for $A_0 = 0$ ($\chi^2 = 30.7$, $Q = 0.72$) and much stronger regularity for $A_0 = 1$ ($\chi^2 = 1455.0$, $Q \approx 0$); see table 20. Other attempts at refinement failed. Similar results have been reached for the other equation. Refinement of spurious regularities between A_0, A_2, and A_4 also failed.

Table 20: CONTINGENCY-all regularity between A_3 and A_4 for slices of A_0

$A_0 = 0$, $\chi^2 = 30.7$, $Q = 0.72$

Attribute A_4							
6	10	13	12	10	8	11	9
5	10	15	6	11	5	13	11
4	1	11	11	11	13	11	5
3	15	16	13	13	11	12	8
2	6	15	8	12	16	9	11
1	13	11	9	7	10	12	6
0	8	13	13	10	12	12	6
	1	2	3	4	5	6	7

Attribute A_3

$A_0 = 1$, $\chi^2 = 1455.0$, $Q \approx 0$

Attribute A_4							
6	0	0	0	0	0	0	9
5	0	0	0	0	60	67	55
4	0	0	71	67	2	0	0
3	0	82	3	0	0	0	0
2	39	1	0	0	0	0	0
1	30	0	0	0	0	0	0
0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7

Attribute A_3

9 Application of 49er to Navy Training Database

Persons enlisted in the Navy are usually promised training in a skill. A person can get such a training by attending an "A" school. S/he may attend the "A" school immediately after boot-camp or may first serve on a ship and then go to school.

Some "A" schools have very high attrition rates – over 30%. Attrition is costly to the Navy because it looses the investment in the individual and the opportunity of investing in another individual. Attrition has a human cost – the individual is branded as a failure and is probably more likely to attrite from the Navy or cause discipline problems.

Two data files have been combined to provide data for the analysis of attrition: the Survival Tracking File and the Training Tracking File. The first contains basic data about

an individual: paygrade, duty, and various personal information. The second cont[illegible] data about every training incident that the individual has had, including start date, end date, attrition, type of training, etc. Data starts in 1979.

9.1 Strategy of search

We used 49er to search for regularities in the Navy Training data file that may indicate the reasons for the high attrition rate in the Naval training program. The OUTCOME attribute has been selected as the single dependent attribute and was compared to all other attributes. Regularities were sought for all data as well as for slices.

Because of the large number of attributes (35) the search which considers all combinations of data slices would be very costly.

The original Navy file was broken down initially into 3 data subsets:

- background.dat
- event.dat
- scores.dat

In each of these subsets the dependent attribute OUTCOME was compared to all other attributes in the data set looking for regularities that might lead to explanations of the known outcomes.

The attributes in event.dat include:

```
Outcome  vs     Length of Service (Months)
                Length in Pipeline (Days)
                Days Awaiting Instruction
                Interrupted Instruction Days
                Under Instruction Days
                Number of Academic Setbacks
                Number of Non Academic Setbacks
                Number Of Accelerations
                Number of Interruptions
```

The number of values for some attributes (Length of Service, Length in Pipeline, Under Instruction Days, Days Awaiting Transfer, and Days Awaiting Instruction) was too large for useful interpretation. These values were aggregated into groups of 10 days in the CONTINGENCY-2 and CONTINGENCY-all tables.

The attributes in scores.dat include different test results:

```
Outcome  vs     AFQT SCORES
                AR  (arithmatic reasoning)
                EI  (electronics information)
                GS  (general science)
                MC  (mechanical comprehension)
                MK  (mathematical knowledge)
                NO  (numerical operations)
                WK  (word knowledge)
```

The attributes in background.dat include:

```
Outcome  vs     sex indicator
                race indicator
                age
                cohort
                paygrade
                soc
                ed_cert
                ed_yrs
                HSGDIP
                waiver
                loss_date
```

Initially, 49er has been used to search for CONTINGENCY-2 and CONTINGENCY-all regularities and equations in these three subsets of data. After the results were analyzed, several new questions led to further applications of 49er. The results are summarized in this report.

9.2 Summary of results; STEP I

Goal: Run 49er on each of the three data sets to determine possible causes for high attrition rate.
Parameters: The OUTCOME attribute was set as the dependent attribute against all other attributes in the dataset.

Results for event.dat
The results show many interesting regularities.

1. Passing is strongly dependent on the Number of Days Under Instruction. The more days of instruction a student received the higher was the rate of Passing. That is obvious, but the CONTINGENCY-all tables provide interesting details.

2. Passing is strongly dependent on the number of Academic Setbacks. Those with fewer Academic Setbacks passed at greater rates. A weaker regularity showed that one or more of Accelerations also positively affected passing rates.

3. Non Academic Setbacks affected the Outcome in different way depending on Days Awaiting Transfer, Number of Interruptions, and Length of Interruptions. When Days Awaiting Transfer was less than 10 or the Number of Interruptions was zero, more than zero of Non Academic Setbacks improved the chances of passing. Perhaps those who stayed longer in the training have had a higher chance to experience Non Academic Setbacks than many of those who dropped. However, when Days Awaiting Transfer was more than 10 or the Number of Interruptions was more than zero, the increasing number of Non Academic Setbacks decreased the chances of Passing, so the previous explanation does not work for those cases. Perhaps when different obstacles accumulate, the chances of Passing strongly decrease.

4. Academic Attrition was affected by Academic Setbacks, Length of Service, and Under Instruction Days or Length in the Pipeline.

   - For more than zero Academic Setbacks, Academic Attrition is greater than expected.
   - Academic Attrities become greater than expected when Days Under Instruction or Length in the Pipeline is less than 90 days.
   - A very interesting fact appeared concerning length of service. The actual frequency tables show that 2/3 of all students receive training right after boot camp or within the first 10 months of service. The results show that the proportion of Academic Attrites in this group was greater than for students with more than 10 months of service. Students with more than 10 months of service, however, show a greater than expected Non-Academic Attrition rate.

Results for scores.dat

The analysis performed on the Scores.dat data file for Outcome as dependent attribute did not show any strong regularities between the test scores and Passing or Attrition. This suggests that passing rates are not affected by students' test scores.

Results for background.dat

Several interesting regularities were found in comparing Background Data with Outcome:

1. Passing rates depend on Sex. Male students passed at greater rates than females.
2. Outcome was also effected by Race. Whites experienced greater than expected passing rates than Non-Whites.
3. No strong regularities was found between ED_Yrs and Outcome. The majority of students did have 12 years of education.

9.3 Summary of results; STEP II

Goal: Since Academic Setbacks and Under Instruction Days were critical in determining Outcome, analyze what factors affect Academic Setbacks and Under Instruction Days.
Parameters I: A new data set, including test scores, ed_cert, ed_yrs, HSGDIP was run against the dependent attribute Academic Setbacks.
Results: No strong regularities were found that would support a definitive contributing factor for Academic Setbacks in this data set.

Parameters II: Academic Setbacks and Days Under Instruction were set as the dependent attributes against the Event.dat data.
Results: The regularities found in this search show several interesting tendencies.

1. Academic Setbacks are effected by the number and Days of Interruptions. As Interruptions become positive and the number of Interrupted days becomes greater than 10, the number of Academic Setbacks is greater than expected.

2. Academic Setbacks were greater than expected when the Days Under Instruction or Length in the Pipeline was greater than 90. One explanation for this can be that students who attrite earlier have less chance of experiencing Academic Setbacks

3. When Days Awaiting Transfer became more than 10, Academic Setbacks there were greater than expected.

9.4 Summary of results; STEP III

Goal: Create a new data set to determine what background factors may contribute to Academic Setbacks or Days Under Instruction.
Parameters: A new Data set was created that combined attributes from Background.dat and Event.dat.
Results: Several very interesting regularities were discovered.

1. Race had major effect on Outcome, Academic Setbacks, and Days Under Instruction. In comparison to Race not= "C", Race = "C" had:

   - Greater than expected Passing rates
   - Less than expected Academic Setbacks
   - Greater than expected Days of Instruction = 90 days or more

2. Sex had showed similar results. In comparison with females, males had:

   - Greater than expected Passing rates
   - Less than expected Academic Setbacks
   - Greater than expected: Days Under Instruction $\geq$ 90
   - Greater than expected Non-Academic Setbacks

3. SOC (student origin code) shows the following regularities:

   - Students whose Origin code = "B" - (basic training/boot camp) showed a greater than expected rate of Days Under Instruction LESS than 90 days.
   - Students Whose Origin Code NOT = "B" had a greater than expected rate of Days Under Instruction GREATER than 90 days.

4. COHORT year shows that the students entering the program in 1983 and earlier, in comparison to the students who entered the program after 1983, had:

   - Greater than expected Academic Attrites
   - Greater than expected Academic Setbacks
   - Greater than expected: Days Under Instruction LESS than 90
   - Less than expected Non-Academic Setbacks
   - Less than expected Interruptions

9.5 Examples of results

In this section we illustrate some of the results discussed earlier in this section on contingency tables from which they were inferred. We also show examples of regularity refinement. These are only a few selected examples from a very large number of regularities detected in the Navy Training Database.

event.dat

Passing strongly dependents on the Number of Academic Setbacks. Students who had 0 Academic Setbacks passed at greater rates, while other students became attrites at greater rates. This conclusion was inferred from the following CONTINGENCY-ALL regularity for OUTCOME and ACSB (Academic setbacks). PASS means 'passing the training', NON_AC means 'non-academic attrites', ACADEM means 'academic attrites'. In the bottom row, 0, 1, 2, 3, 5 indicate the number of academic setbacks. In the following difference table positive numbers in the table indicate events which occur more frequently that expected based on the null hypothesis of independence between OUTCOME and Academic Setbacks. For instance, in the first column, positive numbers indicate that trainees with zero Academic Setbacks pass or become non-academic attrites at the greater than expected rate, while they are academic attrites at a less than expected rate. Trainees with one or more Academic Setbacks are academic attrites at a greater than expected rate, as indicated in the ACADEM row of the table. The value of -1 means no data in a given cell. As explained in Section 5, 49er considers several measures of regularity strength. Their values are listed under the corresponding tables below.

```
Attributes: ACADEMIC SETBACKS VS OUTCOME
Range:      All 6032 records
```

OUTCOME

PASS	0.0148	-0.0589	0.0500	0.0487	0.5294
NON-AC	0.1800	-0.4343	-0.6518	-1	-1
ACADEM	-0.1024	0.3064	0.1050	0.2326	-1
	0	1	2	3	5

ACSB

$\chi^2 = 99.35$

Probability of random fluctuation $Q = 5.79 \cdot 10^{-18}$

Cramer's $V = 0.09$

Contingency coefficient $C = 0.13$

background.dat

OUTCOME versus SEX and RACE

OUTCOME

PASS	-0.0653	0.0327
NON-AC	-0.1710	0.0855
ACADEM	0.2287	-0.1144
	FEMALE	MALE

SEX

The positive and negative differences in this table show that males, compared to females, are academic attrites at a smaller rate, while pass or become non-academic attrites at a greater rate. The probability of random fluctuation Q = 4.91e-13.

The next table shows the relationship between RACE and OUTCOME. The positive and negative differences in this table show different patterns of dependence between OUTCOME and RACE. The frequency table (not reproduced in the report) indicates that only for the race 'Z' there was not enough data (just two records) to make a conclusion. For this table, the probability of random fluctuation Q = 9.58e-11.

OUTCOME

PASS	0.0250	0.0196	-0.1699	-0.0168	-0.0267	-0.2353
NON-AC	0.0114	-1	0.0119	-1	-0.4018	4.4836
ACADEM	-0.0680	0.3073	0.4315	0.4007	0.2123	-1
	C	M	N	R	X	Z

RACE

A weaker but interesting CONTINGENCY-ALL regularity holds for Academic Setbacks (ACSB) vs. Non Academic Setbacks (NASB).

ACSB

5	0.1292	-1	-1	-1	-1
3	-0.0644	0.4362	1.3937	-1	-1
2	-0.0635	0.3564	1.3272	1.9921	-1
1	-0.0489	0.3409	0.6669	0.6669	-1
0	0.0193	-0.1292	-0.2934	-0.3125	0.3750
	0	1	2	3	4

NASB

$\chi^2 = 56.346687$

Probability of random fluctuation $Q = 2.1326926 \cdot 10^{-6}$

Cramer's $V = 0.048325185$

Contingency coefficient $C = 0.09620209$

This regularity reveals a simple pattern: those who have got setbacks of one type (academic or non-academic), show a tendency to have more than expected setbacks of the other type. This pattern is very well captured by CONTINGENCY-2 regularity reproduced below, which is stronger than CONTINGENCY-ALL (probability of 10^{-11} vs. 2×10^{-6}). In the difference table below, the positive numbers 0.019 and 0.398 show the positive correlation between both types of setback.

ACSB	Differences		Actual counts		Expected counts	
> 0	-0.051	0.398	1382.00	263.00	1456.83	188.17
0	0.019	-0.149	3960.00	427.00	3885.17	501.83
	0	> 0	0	> 0	0	> 0

NASB

$\chi^2 = 46.2$

Probability of random fluctuation $Q = 1.07 \cdot 10^{-11}$

Cramer's $V = 0.088$

Contingency coefficient $C = 0.087$

We reproduced all three tables important for reasoning about contingency tables and defined in section 5: difference table, actual frequency table, expected frequency table. 49er returns all three, because all three are needed to make sound conclusions. To save space, we normally report only the difference table of the actual frequency table.

An example of regularity refinement

Consider the refinement process applied to the CONTINGENCY-2 for the attributes Academic Setbacks vs Under Instruction Days. The CONTINGENCY-2 refinement caused a shift of the split point in the CONTINGENCY-2 table from 90 Days Under Instruction to 100 Days Under Instruction. The Chi-square and Cramer coefficient values increased very significantly, providing evidence that dividing data at 100 days under instruction is more significant than 90 days. Details of the refinement follow:

Before refinement

UIDAYS		
> 9	-0.189	0.504
≤ 9	0.131	-0.350
	≤ 0	> 0

ACSB

After refinement

UIDAYS		
> 10	-0.538	1.434
≤ 10	0.079	-0.211
	≤ 0	> 0

ACSB

	Before	After
$\chi^2 =$	398.88	683.18
Probabil. of random fluct. $Q =$	0.0	0.0
Cramer's $V =$	0.26	0.34
Contingency coefficient $C =$	0.25	0.32

Both patterns are very strong, as indicated by (almost) zero probability of random fluctuation. In this table, UIDAYS are related to the number of Academic Setbacks. UIDAYS has been also refined to 100 days for several other attributes. For some other attributes, such as Awaiting Transfer Days and INTERRUPTIONS, however, a "natural" split point obtained in the refinement process is at 70 days. This may mean that 70 Days and 100 Days Under Instruction are significant for different reasons in relation to different attributes.

10 Application of 49er to Navy Recruitment Database

NPRDC did a survey of Navy recruiters to find out how they like their job. Recruiters have to make a quota every month. Since the number of recruiters is dropping, mostly for demographic reasons, NPRDC wants to know how the recruiter's job can be made better.

10.1 Strategy of search

Two natural outcome variables for the task are SATISFY and SUCCESS. They measure whether the recruiter is satisfied with his job and how successful he is at the job. We used SATISFY and SUCCESS as dependent variables, and all other variables as independent, to find those with highest influence on SATISFY and SUCCESS.

10.2 Summary of results for SATISFY

A number of strong regularities have been detected, which have been expressed as equations.

Train vs. Satisfy: TRAIN is the training scale. 49er came up with a number of equations which describe these data. The best fit has been $Y = (\log(A + B \times X))$, where Y is the dependent variable (SATISFY), whereas X is independent (TRAIN). $A = 2.71$, with error of A equal 0.35; $B = 0.75$ with error of B equal 0.11. Tracing the positive values from the lower left corner to the upper right corner in the following difference table produced by 49er, one can get a rough estimate of this equation. In this table, the values of TRAIN (training scale) from 10 to 68 have been aggregated in 10s (1 – 7).

Attribute: SATISFY (#1)

7	-1	-.6499	-.4939	-.0566	.92768	4.956	11.905
6	-1	-.7357	-.4555	.02256	1.2199	2.3431	1.6226
5	-.3091	-.5916	-.2142	0.1798	.53007	.54368	-1
4	-.8334	-.4978	-.0154	.23383	.15962	-.3858	-1
3	-.4923	0.0461	.17374	0.032	-.2381	-.7569	.10612
2	1.2979	.77479	.26646	-.2477	-.7189	-.7433	-1
1	2.3983	1.2849	.07595	-0.346	-.7043	-1	-1
	1	2	3	4	5	6	7

Attribute: TRAIN (#19)

Chi-square = 784.9994

```
      Number of degrees of freedom = 49
Probability of random fluctuation Q = 0.0
                        Cramer's V = 0.18569934
          Contingency coefficient C = 0.44096622
```

TEAMWORK vs. SATISFY: The best equation for these attributes has the same form $Y = (\log(A + B \times X))$ as one for TRAIN and SATISFY, but different parameter values $(AB) = (3.3012410.44533682)$ and different errors for A and B (0.29477584 0.07523044).

Many other regularities for SATISFY have been detected in the form of similar equations for **SUPPORT, SUPERS (supervisor quality), Job Stress, RESPECT, Personal Support,** and several other attributes.

Hours vs Satisfy: Recruiters had a greater than expected level of job satisfaction when the value for hours was less than or equal to 3. Job dissatisfaction was greater than expected when the value for hours became greater than 3.

Freeman vs Satisfy: Job satisfaction was greater than expected for recruiters who had not been nominated for a freeman transfer. Recruiters who had been nominated showed greater than expected rates of job dissatisfaction.

Individual 6 Month Goals vs Satisfaction: Recruiter who made their goal at least 5-6 times within the past 6 months had a greater than expected rate of job satisfaction. Recruiters who did not achieve their goal at least 5 times had a greater than expected rate of job dissatisfaction.

Station goals vs Satisfaction: Station goals were consistent with individual goals. Recruiters from stations which made their goal at least 5 times in the last 6 months had a greater than expected level of job satisfaction, while those from stations which did not achieve their goals had a greater than expected level of job dis-satisfaction.

Length as Recruiter vs Satisfy: An interesting CONTINGENCY-ALL regularity. Recruiters who were new to the job showed a greater than expected rate of job satisfaction. Recruiters who had been on the job for 3 years or more also showed a significantly greater than expected rate of job satisfaction. Recruiter on the job between 9 and 36 months showed a greater than expected rate of job dis-satisfaction.

This is interesting because success rate of new recruiters is very low. A recruiter usually becomes successful after 9 months on the job. This regularity proves that success is not the necessary factor in job satisfaction.

$$\begin{aligned} \chi^2 &= 495.04 \\ \text{Probability of random fluctuation } Q &= 1.34 \cdot 10^{-18} \\ \text{Cramer's } V &= 0.14 \\ \text{Contingency coefficient } C &= 0.36 \end{aligned}$$

10.3 Summary of results for SUCCESS

Leave (C19) vs Success: If leave was limited to not more than 10 days, there was a greater than expected rate of 100% success in meeting the goals. If the number of days of leave was greater than 20 there was a much greater than expected rate of low, 0 – 33%, success.

The CONTINGENCY-2 refinement shows that the dependence of SUCCESS on the LENGTH of LEAVE is very significant. When we compare recruiters who take more than 20 days leave with those who take 20 days or less, the rate of 67% or more success is very significantly greater for those who take less leave.

	CONT-2	CONT-2 Refinement	CONT-ALL
chi^2 =	33.11	173.13	357.26
Probability of random fluctuation Q =	$8.71 \cdot 10^{-9}$	$1.5 \cdot 10^{-39}$	0.0
Cramer's V =	0.10	0.23	0.17
Contingency coefficient C =	0.10	0.22	0.31

Advance vs Success: A very interesting CONTINGENCY-ALL regularity. A greater than expected number of recruiters with a success rate of 67% or more felt that they had advanced slower than average. Recruiters with a poorer performance felt that their careers had advanced somewhat faster than average.

χ^2 =	134.32
Probability of random fluctuation Q =	$5.99 \cdot 10^{-19}$
Cramer's V =	0.10
Contingency coefficient C =	0.20

Freeman vs Success: A much greater than expected number of recruiters who reached there goals 67% or less of the time and were nominated for a freeman transfer within the past year. There was a much greater than expected number of recruiters who did not know if they were nominated or not.

	CONT-2	CONT-2 Refinement	CONT-ALL
χ^2 =	26.39	80.51	414.9
Probability of random fluctuation Q =	$2.79 \cdot 10^{-7}$	$2.9 \cdot 10^{-19}$	0.0
Cramer's V =	0.09	0.16	0.20
Contingency coefficient C =	0.09	0.16	0.33

Time At Present Station vs Success: The longer a recruiter is at a station the more success s/he will experience. In addition to strong CONTINGENCY-2 and CONTINGENCY-ALL regularities, a linear dependence of SUCCESS on Time At Present Station has been also discovered.

	CONTINGENCY-2	CONTINGENCY-ALL
χ^2 =	174.65	10009.92
Probability of random fluctuation Q =	$7.12 \cdot 10^{-40}$	0.0
Cramer's V =	0.24	0.28
Contingency coefficient C =	0.23	0.23

```
Linear regularity:  Slope: 0.088  Y-Intercept: 0.222
      Deviations:   dev = 0.449   r^2 = 0.14
```

r^2 is the square of the correlation coefficient. *dev* measures the difference between predicted and actual values of the dependent variable.

Time as a Recruiter vs Success: The longer an individual serves as a recruiter the more successful s/he will be. Regularities in all categories have been detected for these attributes:

	CONT-2	CONT-2 Refinement	CONT-ALL
χ^2 =	137.31	375.48	1652.28
Probability of random fluctuation Q =	$1.033 \cdot 10^{-31}$	0.0	0.0
Cramer's V =	0.21	0.34	0.36
Contingency coefficient C =	0.21	0.32	0.58

```
Linear regularity:  Slope: 0.103  Y-Intercept: 0.062
      Deviations:   dev = 0.436   r^2 = 0.189
```

Community vs Success: An interesting CONTINGENCY-ALL regularity. Recruiters from the surface communities had a less than expected rate of achieving 100% of their goals. Recruiters from air and other communities had a significantly higher than expected rate of achieving their goals 100% of the time. Recruiters from the medical community had a much greater than expected rate of achieving their goals only 33% of the time.

χ^2 =	152.24
Probability of random fluctuation Q =	$1.27 \cdot 10^{-30}$
Cramer's V =	0.11
Contingency coefficient C =	0.21

Time in the Navy vs Success: Length of service influences positively the success rate of recruiters (CONTINGENCY-ALL regularity).

χ^2 =	201.74
Probability of random fluctuation Q =	$4.81 \cdot 10^{-30}$
Cramer's V =	0.12
Contingency coefficient C =	0.24

Paygrade vs Success: A similar but weaker CONTINGENCY-ALL regularity has been found between PAYGRADE and SUCCESS. This is probably related more to the time in service than paygrade.

χ^2 =	101.89
Probability of random fluctuation Q =	$1.53 \cdot 10^{-14}$
Cramer's V =	0.09
Contingency coefficient C =	0.17

11 Conclusions

Forty-Niner has been able to discover statistically significant regularities in various real-world databases, but whether they may have practical applications depends on the users, who must interpret and apply the results. However, more databases must be tried before we can conclude usefulness of our exploration paradigm stated in Section 1. The answer is only partially the test of 49er. It also depends on the nature of databases and can be different for databases in different domains.

12 Acknowledgments and Disclaimer

The opinions expressed in this paper are those of the author, are not official, and do not necessarily reflect the views of the Navy Department. The work described in this paper was supported by the Office of Naval research under the grant No. N00014-91-J-1362. We would like to thank Susan Chipman, David Krantz, Rae Silver, Steve Sorensen, and Gregory Piatetsky-Shapiro for their advice, and/or their databases and details of their data analysis. Special thanks to Drs. Susan Chipman and Steven Sorensen for their inspiration, many helpful discussions and suggestions.

13 Publications sponsored by this grant

Integration of Knowledge and Method in Real-World Discovery, SIGART, 1991.

The KDD Land of Plenty, in Piatetsky-Shapiro, G. ed. Proceedings of the AAAI-91 Workshop on Knowledge Discovery in Databases, Anaheim, July 1991, p.iii-vi.

Automated Empirical Discovery in a Numerical Space (with Zhu, J), in the Proceedings of the Third Annual Chinese Machine Learning Workshop, July 15-19, 1991, Harbin Institute of Technology, p.1-11.

Human Discovery of Laws and Concepts; An Experiment (with Zytkow, A), in Proceedings of the 13th Cognitive Science Conference, Lawrence Earlbaum Associates, 1991, p.617-622.

Automated Discovery of Empirical Equations from Data (with Zembowicz, R), in Ras Z. and Zemankova M. (eds.) Methodologies for Intelligent Systems, Springer-Verlag, 1991, p.429-440.

Control of Automated Empirical Discovery by Diagrammatic Representation of Theory (with Zhu, J.), in Narayanan H. ed. Working Notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations, Palo Alto, CA, March 1992. 176-179.

The First Phase of Real-World Discovery: Determining Repeatability and Error of Experiments (with Zhu, J. and Zembowicz, R.), in Machine Learning: Proceedings of the Ninth International Conference, July 1992, Aberdeen, United Kingdom. Morgan Kaufmann Publ.

Operational Definition Refinement: a Discovery Process (with Zhu, J., and Zembowicz R.). Accepted for the Proceedings of the Tenth National Conference on Artificial Intelligence, The AAAI Press, 1992.

Discovery of Equations: Experimental Evaluation of Convergence (with Zembowicz R.). Accepted for the Proceedings of the Tenth National Conference on Artificial Intelligence, The AAAI Press, 1992.

A Graph Representation of an Empirical Theory: Guiding a Machine Discovery Process (with Zembowicz, R. and Zhu, J.), in Kishore S. ed. Proceedings of AAAI-92 Workshop on Communicating Scientific and Technical Knowledge, July 1992. To be published.

Discovery of Regularities in Databases (with Zembowicz, R.), in Zytkow J. ed Proceedings of the ML-92 Workshop on Machine Discovery. July 1992, Aberdeen, U.K. 18-27.

14 References

Bhattacharyya, G.K. & Johnson, R.A. (1986) *Statistical Concepts and Methods*, New York:NY: Wiley.

Cai, Y., Cercone, Y., and Jiawei, H. (1989). Attribute-oriented Induction in Relational Databases, *Proceedings of The International Workshop on Knowledge Discovery in Databases.*

Chimenti, D. *et al.* (1990). The LDL System Prototype, *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.1

Chipman, S.F., Krants, D.H., Silver, R. (1990). Mathematics Anxiety and Science Careers Among Able College Women, *Technical Report*

Eadie, W.T., Drijard, D., James, F.E., Roos, M., Sadoulet, B. (1971) *Statistical Methods in Experimental Physics*, North-Holland Publishing Company.

Glymour, C., Scheines, R., Spirtes, P., Kelly, K. (1987). *Discovering Causal Structure*, Academic Press, San Diego, California.

Glymour, C., Scheines, R., Spirtes, P., Kelly, K. (1991). *Causality, Statistics and Search*, Technical Report, Department of Philosophy, Carnegie Mellon University.

Gokhale, D.V. & Kullback, S. (1978) *The Information in Contingency Tables*, Marcel Dekker, Inc. New York, NY.

Langley, P., Simon, H. A., Bradshaw, G. L. & Żytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes.* Cambridge, MA: MIT Press.

Lisp-Stat: Book Review (1991) *Statistical Science*, Vol.6, No.4, 339-362.

Naqvi, S., and Tsur S., (1989). *A Logical Language for Data and Knowledge Bases*, Computer Science Press, New York.

Piatetsky-Shapiro, G. ed. (1991) Proc. of AAAI-91 Workshop *Knowledge Discovery in Databases*

Piatetsky-Shapiro, Frawley ed. (1991). *Knowledge Discovery in Databases*, Menlo Park, Calif.: AAAI Press.

Press, W. H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. (1989) *Numerical Recipes in Pascal*, Cambridge: Cambridge Univ. Press.

Piatetsky-Shapiro, G. and C. Matheus, (1991) Knowledge Discovery Workbench, in: G. Piatetsky-Shapiro ed. Proc. of AAAI-91 Workshop *Knowledge Discovery in Databases*, 11-24

Shrager, J. and Langley, P. eds (1990). Computational Models of Scientific Discovery and Theory Formation, Morgan Kaufmann, San Mateo, CA.

SPSS Inc., (1990) *SPSS Reference Guide*, Chicago, IL.

Tierney, L. (1990). *Lisp-Stat: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*, Wiley & Sons.

Zembowicz, R., Żytkow, J.M. (1991). Automated Discovery of Empirical Equations from Data, *Proceedings of the ISMIS-91 Symposium*, Springer-Verlag

Żytkow, J.M. (1987). Combining many searches in the FAHRENHEIT discovery system, *Proc. 4th International Workshop on Machine Learning*, Morgan Kaufmann, Irvine, CA. 281-287.

Żytkow, J., and Baker, J., (1991). Interactive Mining of Regularities in Databases, in: *Knowledge Discovery in Databases*, eds. G. Piatetsky-Shapiro and W. Frawley. Menlo Park, Calif.: AAAI Press.